НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему “Мульти графічна візуалізація числових даних та їх обмін між користувачами “

Виконав (-ла): студент (-ка) ___4___курсу, групи_ТІ-51_

Скочко Богдан Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доктор технічних наук, професор Бадаєв Ю. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)

”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім’я, по батькові)

1. Тема роботи _____“Мульти графічна візуалізація числових даних та їх обмін між користувачами“_____
_____

керівник роботи_____Бадаєв Юрій Іванович, доктор технічних наук, професор
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”____ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _____Форма реалізації – веб-застосунок з інтерфейсом для користувача (HTML5, CSS, Angular, PostgreSQL). Середовище розробки програмного продукту – Visual Studio Code._____

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)____Проаналізувати основні проблеми візуалізації числових даних у різних форматах, дослідити існуючі програмні засоби. Розробити веб-застосунок, що буде надавати користувачам можливість будувати діаграми різних типів для своїх даних та ділитися ними з іншими користувачами._____
_____
_____

Київ – 2019

5. Перелік ілюстративного матеріалу
___«Мета роботи», «Постановка задачі», «Архітектура програмного застосунку», «Функції розробленого програмного застосунку», «Схема роботи користувача з програмою», «Функції серверної частини програми», «Функції клієнтської частини програми», «Архітектура клієнтської частини», «Використані технології», «Графічний інтерфейс користувача», «Висновки»_____
_____
_____
_____

6. Дата видачі завдання "_10_"___жовтня_____2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.2018-31.03.2019	
2	Розробка архітектури та загальної структури системи	01-21.04.2019	
3.	Розробка структур окремих підсистем	22-28.04.2019	
4.	Програмна реалізація системи	29.04-13.05.2019	
5.	Оформлення пояснювальної записки	06.05-01.06.2019	
6.	Захист програмного продукту	25.05.2019	
7.	Передзахист	01.06.2019	
8.	Захист	17.06.2019	

Студент _____ Скочко Б. О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Бадаєв Ю. І.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення ефективної web-реалізації проблеми візуалізації числових даних.

Об’єктом дослідження є типи діаграм та графіків, що можуть бути використані. Було проаналізовано існуючі програмні системи, виявлено їх недоліки. Створено веб-систему, що дозволяє користувачу створювати та налаштовувати діаграми різних типів для своїх даних та ділитися ними з іншими користувачами.

Створена програмна система може бути використана для широкого спектру завдань графічної візуалізації.

Загальний обсяг роботи: 54 сторінки, 24 ілюстрації, 8 бібліографічних посилань та 3 додатки.

Ключові слова: стовпчикові діаграми, секторні діаграми, точкові діаграми, діаграми потоків, візуалізація числових даних.

ABSTRACT

The goal of the work is to create an effective web implementation of numerical data visualization.

Different diagram types that can be used are the object of study. Review of existing systems was conducted with their disadvantages summed up. Web system that allows a user to create and share different types of diagrams was created.

Developed software system can be used in wide range of visualization applications.

Total volume of the paper: 54 pages, 24 illustrations, 8 bibliography links and 3 appendixes.

Keywords: bar charts, pie charts, scatter plot, data flow diagrams, numerical data visualization.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1) JSON — JavaScript Object Notation; подання даних з використанням синтаксису мови JavaScript.

2) CSV — Coma Separated Values; подання даних у неявному табличному вигляді, коли окремі значення відділені роздільником (наприклад комою).

3) Візуалізація — процес перетворення даних (або уявлення про дані) до графічного формату, який може бути легко сприйнятий людиною.

4) Діаграма — функціональна одиниця візуалізації, що має конкретні характеристики розподілу даних.

ЗМІСТ

ВСТУП

Для сучасного світу є характерним збільшення попиту на новітні технології візуалізації даних різних форматів.

Інформація, зображена у вигляді графіків, складних таблиць, гістограм, кругових діаграм тощо, дає змогу проведення кращого аналізу, оцінювання своїх властивостей, виділення складових частин і закономірностей. Дані часто утворюють статистичні розподіли та ілюструють різноманітні тенденції, які важко помітити, якщо вони представлені у примітивному текстовому, списковому або табличному форматі.

Без сумніву, така візуалізація слугує потребам людини, а не машини, таким чином відбуваючись як сценарій людино-машинної взаємодії.

У переважній більшості застосунків програмна система отримує у якості вхідних даних файл, призначений для машинної обробки, вміст якого не може бути розпізнаний людиною без залучення спеціалізованих засобів. Вихідними даними роботи системи постають чіткі кольорові зображення різних форм та розмірів відповідно до розробленого дизайну та потреб кінцевого споживача.

Візуалізація передбачає роботу з відформатованою певним чином інформацією, певні закони та особливості структури якої зазвичай добре відомі та повинні бути підкреслені у ході функціонування програмного засобу, оскільки це є одним з його головних завдань.

Різні предметні області оперують своїми власними технологіями представлення інформації у графічній формі. Як показує практика впровадження, задача візуалізації повинна бути орієнтована на кінцевих користувачів та не може розглядатися без контексту.

Без сумніву, сучасний світ програмних засобів природньо орієнтований на тісну комунікацію у вигляді мережі Інтернет.

Використання можливостей Інтернету дозволяє полегшувати обмін інформацією

між різноманітними джерелами. Засоби, що працюють безпосередньо у web-просторі, слугують медіаторами, оброблювачами, конверторами та транспортувальниками даних різних форматів.

Подібний підхід сприяв появі великої кількості стандартів. Консорціум WWW, який займається розвитком та уніфікацією web-технологій, розробив стандарти для роботи зі статичними та динамічними даними різноманітних предметних областей, в тому числі для візуалізації певної інформації для широкого спектру потенційних цілей використання кінцевого продукту.

Сучасний світ програмного забезпечення фактично складається зі стандартів та інтерфейсів взаємодії з ними, які утилізують (активно використовують) web-орієнтовані системи, кінцевою метою яких є задоволення певної гострої потреби, що виникає у кінцевого користувача.

Візуалізація web-засобами спирається на існуючі моделі, що використовувалися протягом багатьох тисячоліть науковцями та митцями для витворення зручного для сприйняття образу з описового або чітко заданого текстового формату. Прагнення людства до перенесення інформації у малюнки, що описують її структуру та характер розподілу, було зафіксовано ще на ранніх етапах розвитку світової культури, зокрема у славнозвісних наскальних малюнках.

Винайдення цифр та чисел сприяло активному розвитку числення як математичної науки, утворенню абстрактного шару, який, у свою чергу, потребував відповіді у більш давньому форматі зображення, що призвело до появи своєрідної візуалізації метаданих про кількість або числових даних, що пройшли попередню обробку засобами абстрактного обчислення. Так мова йшла вже не про “3 людини та 2 дерева”, які можна було зобразити лише як фігури живих істот, а про відношення “трьох до двох”, яке можна було зобразити у вигляді діаграми або таблиці.

Подібна взаємодія візуалізації та абстрактного обчислення дала потужний поштовх розвитку прикладних розділів математики, які могла використовувати для свої потреб фізика, наука про закони природи.

Без виконання графічного зображення результатів абстрактних (безпредметних) числових обчислень неможливим би був науково-технічний прогрес у всіх його

проявах, оскільки такого роду візуалізація, без сумніву, означала втілення теоретичних відкриттів у практичні реалії використання.

В наш час візуалізація відіграє ще більшу роль, оскільки обробці підлягають величезні обсяги даних. Зокрема, числових, які використовують статистика та нова наука про дані (data science).

Гігантські корпорації як Google та Facebook створюють власні застосунки для вирішення проблем візуалізації даних про своїх користувачів та навколишнє середовище. Сприйняття цих даних людиною є неможливим у примітивному текстовому форматі. Використання графіків, які узагальнюють отримані результати, значно спрощує виконання аналізу фахівцями предметної області, оскільки їм зазвичай достатньо дуже мало часу, щоб зрозуміти основні тенденції, що характерні для масиву даних, якому відповідає наочне графічне зображення.

1. ЗАДАЧА МУЛЬТИ ГРАФІЧНОЇ ВІЗУАЛІЗАЦІЇ ЧИСЛОВИХ ДАНИХ ТА ЇХ ОБМІНУ МІЖ КОРИСТУВАЧАМИ

Була поставлена задача створення web-системи для побудови графічної візуалізації числових даних у великій кількості форматів та забезпечення ефективної взаємодії між користувачами системи, яка б дозволяла виконувати обмін результатами роботи програми.

1.1 Призначення системи

Основні функції, виконання яких має бути реалізоване:

— побудова різних типів графіків;

— збереження та повторне використання числових та графічних даних;

— імпорт побудованого графіку у форматі HTML для подальшого використання на будь-якому web-сайті;

— забезпечення легкого обміну графічними та числовими даними між різними користувачами;

— можливість забезпечення одночасної роботи одного користувача з багатьма проектами.

Система, що створюється, яка має робочу назву “newChart”, розрахована на легке створення графічної візуалізації числових даних та зручну взаємодію між своїми користувачами.

Сценарій взаємодії користувача з системою показано на рисунку 1.1.

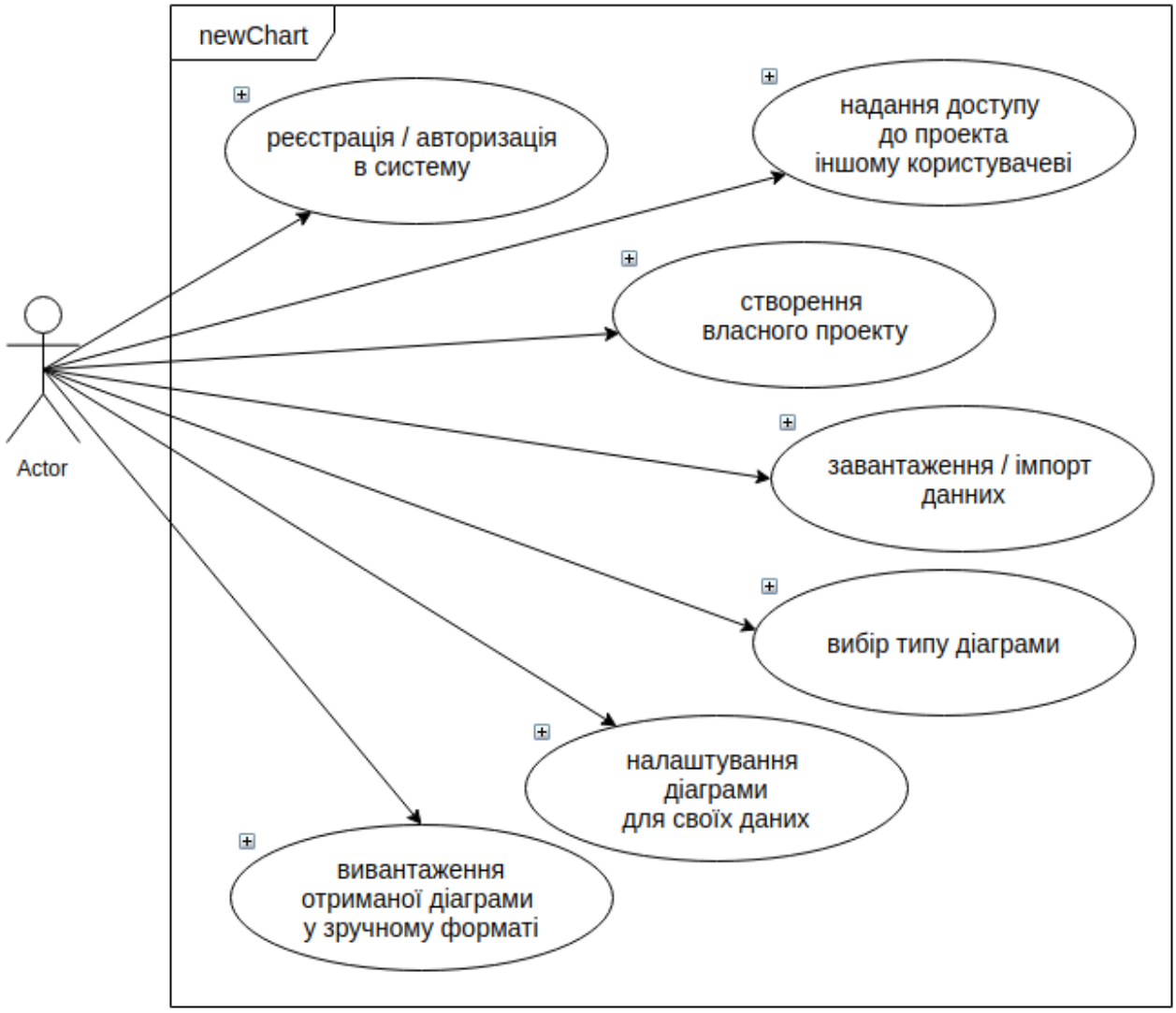


Рисунок 1.1 — Діаграма прецедентів системи

Необхідно реалізувати як прості типи діаграм (точкова, секторна, стовпчикова), так і складніші (наприклад, діаграми потоків).

1.2 Компоненти системи

Програма повинна бути класичним односторінковим застосунком та використовувати перевірені інтерфейси для обміну даними.

Необхідно використовувати сучасні програмні засоби розробки та методи швидкої ефективної обробки значних обсягів числових даних.

Важливим етапом створення системи є вибір архітектури, оскільки на пізніх етапах він не може бути зміненим.

Важливо дотримуватися шаблонних підходів до програмування, оскільки це заощаджує час, впорядковує програмний код, дозволяє писати текст програми, що документує сам себе. Однак це не означає другорядність процесу написання справжньої документації для всіх компонентів застосунку.

Кодова база системи має складатися з двох головних частин:

— клієнтської частини (блок роботи з користувачем),

— серверної частини (блок для обміну та зберігання даних).

1.3 Висновки до розділу

Було описано основні вимоги до системи, яку пропонується розробити, її призначення та очікуваний сценарій дій користувачів по відношенню до неї.

2. АНАЛІЗ ПРОБЛЕМИ МУЛЬТИГРАФІЧНОЇ ВІЗУАЛІЗАЦІЇ ТА ОБМІНУ ДЛЯ ЧИСЛОВИХ ДАНИХ

Для візуалізації та обміну числовими даними існує велика кількість стандартів та форматів, що, без сумніву, підтверджує важливість даної задачі у контексті розвитку сучасної науки про дані.

Деякі з форматів та технологій відрізняються більшою популярністю, однак не існує універсальне рішення для всіх можливих типів проблем.

2.1 Типи графічної візуалізації

Існує декілька найбільш розповсюджених типів подання різноманітних числових даних у вигляді графічних структур.

Оскільки в наш час кожні дані можна подати у числовому вигляді, подібні представлення будуть відрізнятися універсальністю та надзвичайно широким спектром застосування.

Без сумніву, окрім основних, існує безліч специфічних типів подання, які зазвичай краще пристосовані до роботи з інформацією, що стосується окремих предметних областей, однак для вирішення задачі візуалізації, яка не має відношення до окремих типів вхідних даних та однаково ефективно оброблює різноманітні структури, ідеально підходять найбільш універсальні види графіків та діаграм.

Для кожного типу, втім, характерні свої система координат, недоліки та переваги, отже, вони можуть використовуватися як у рамках комплексного підходу, так і самостійно.

2.1.1 Точкова діаграма

Діаграма розсіювання або точкова діаграма (scatter plot) є одним із широко використовуваних типів математичних діаграм.

Дана діаграма використовує декартові координати для відображення значень двох змінних параметрів у певному наборі даних.

Декартова система координат складається з двох осей, що перетинаються під прямим кутом у точці (0, 0), таким чином утворюючи чотири чверті. Кожна вісь має як додатній, так і від'ємний напрями, тобто є коректною числовою прямою.

Дані у точковій діаграмі представлені як список точок, кожна з яких має дві координати, а саме розміщення на горизонтальній та вертикальній осі. Дуже часто тенденцію, характерну для розміщення точок, позначають прямою, рівняння якої легко знайти за допомогою методів лінійної апроксимації (рисунок 2.1).

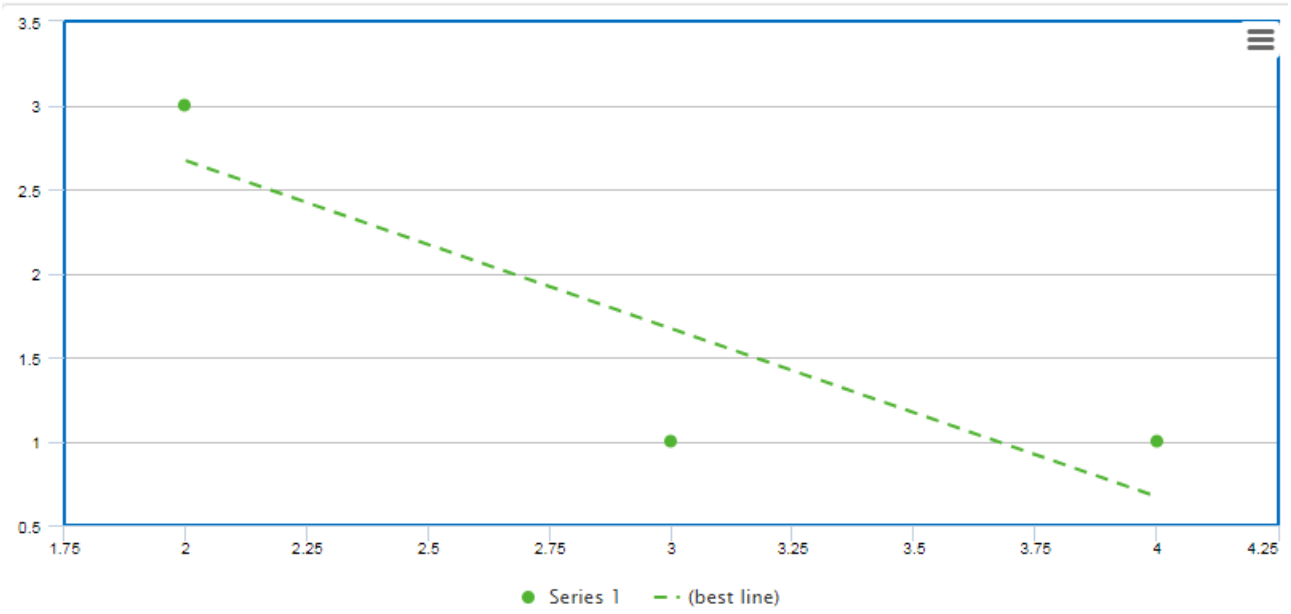


Рисунок 2.1 — Точкова діаграма з лінійною апроксимацією

Кожна з осей може мати довільні мінімальне та максимальне значення. Дані обмеження можна автоматично розраховувати відповідно до мінімуму та максимуму кожної координати у наборі даних.

Кольори, якими позначають точки та осі, в загальному випадку повинні бути близькими та дуже відрізнятися від фонового кольору, що полегшує сприйняття.

Переваги такого типу візуалізації:

— відносна простота побудови,

— реалізація загально-відомого математичного представлення,

— дозволяє виконання класифікації точок відповідно до районів їх скупчень,

— можливість аналізу кореляції між розподілами двох величин,

— існування великої кількості засобів для створення масиву вхідних даних для побудови такого типу зображення.

Недоліки точкових діаграм:

— необхідність розгляду набору даних у контексті двох змінних,

— необхідність вибору різних параметрів відображення (в тому числі, масштабу) для кожної з осей,

— математична модель може бути важкою для сприйняття фахівцями з інших наукових галузей.

Точкові діаграми часто використовуються для демонстрації відношення простої лінійної регресії, яка задається як лінійна функція залежності ординати від абсциси (між двома взаємозалежними параметрами системи), конкретне значення якої є неточним, оскільки модифікується випадковою величиною математичної помилки для кожної пари значень.

Точкові діаграми можна охарактеризувати як відносно простий та ефективний спосіб графічної візуалізації набору даних, що залежить від двох параметрів, який має свої переваги та недоліки.

Існують складніші типи візуалізації, які мають свої переваги для використання їх в особливих умовах предметної області. Точкові діаграми можна охарактеризувати як загальне та найбільш поширене рішення.

2.1.2 Секторна діаграма

Секторні діаграми використовуються для зображення пропорційних складових у виглядів секторів круга.

Під кругом розуміється сукупність усіх складових. Площа сектору, що позначає окрему складову, залежить від її кількісного співвідношення до інших складових.

Кожний сектор є центральним, тому під точкою початку координатного відліку (0, 0) можна розуміти центр круга.

У якості міток секторів часто використовують відсотки, також можуть вказуватися абсолютні значення, якщо це не спричиняє інформаційного перевантаження.

На рисунку 2.2 показано у вигляді секторної діаграми показано результати жартівливого статистичного дослідження, яке проводилося у невеликій вибірці людей з метою встановити яким домашнім тваринам вони надають перевагу. Площа секторів залежить від відсотків респондентів (учасників опитування), які вибрали конкретний варіант.

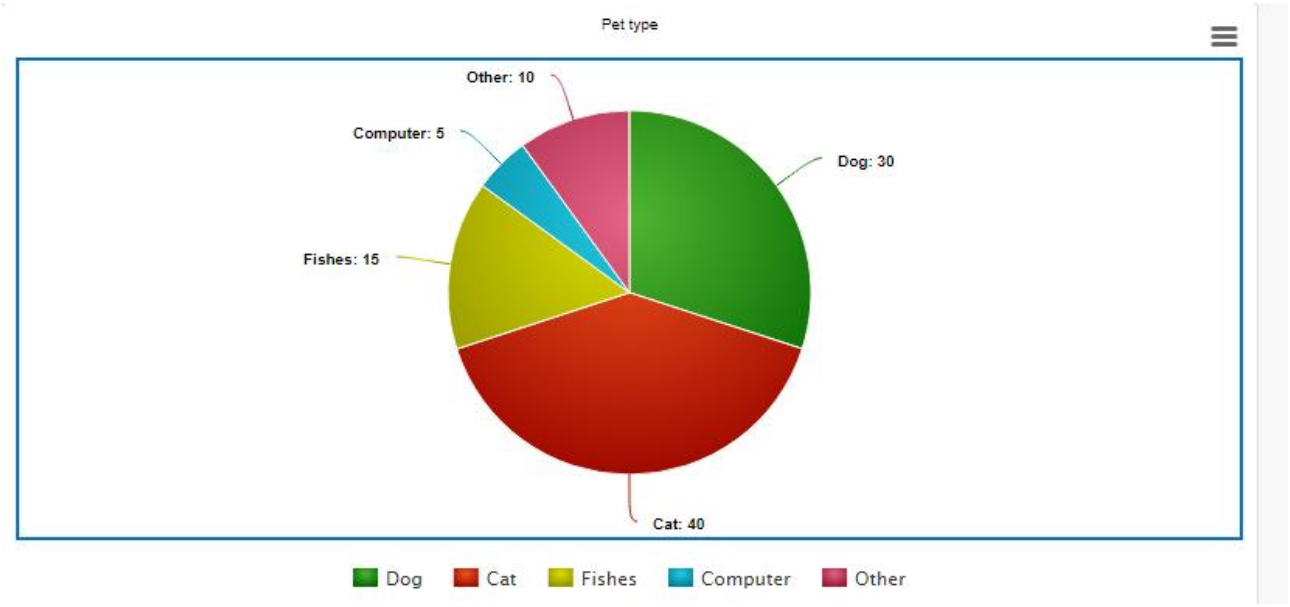


Рисунок 2.2 — Секторна діаграма популярності домашніх тварин

Не відомо, чи було тільки 5 варіантів відповіді, оскільки категорія “Other” може включати в себе декілька або навіть багато варіантів, кожен з яких набрав менше відсотків, ніж найменш популярний з варіантів, зазначених на діаграмі (тобто менше 5%). Цей факт демонструє один з важливих недоліків секторних діаграм, а саме їх неповноту, оскільки коректне зображення усіх варіантів, тим більше тих, що набрали малу кількість відсотків, у вигляді кругових секторів є дуже важкою задачею, бо з точки зору людину суттєво погіршує естетичну привабливість візуалізації.

Переваги секторних діаграм:

— легке рішення для візуалізації пропорційного складу,

— увагу на себе звертають найбільш важливі складники,

— можливість сполучення декількох складників з метою утворення сектору, що буде займати більшу частину круга.

Основні недоліки цього типу візуалізації:

— дані подаються у пропорційному форматі, тому важко уявляти їх абсолютні значення,

— якщо є велика кількість малих складових, їх зображення на крузі буде майже не видно (для цього їх часто сполучають в один),

— легко досягається інформаційне перевантаження, якщо у мітці сектору вказується занадто багато інформації,

— більш пристосовані для поверхневої візуалізації та майже не використовуються у серйозних наукових дослідженнях.

Секторні діаграми дуже часто використовуються для подання кінцевих результатів статистичних досліджень громадськості, оскільки дуже наочно зображують пропорційний розподіл, однак їх проміжне використання майже не зустрічається, бо отримання інформації зі системи координат, в якій вони побудовані, є дуже важким та незручним процесом.

Секторні діаграми не мають за собою складного математичного апарату, оскільки їх реалізація зводиться до кількох тривіальних геометричних формул.

2.1.3 Стовпчикова діаграма

Секторні діаграми використовуються для зображення згрупованих даних у вигляді стовпців (рисунок 2.3). Стовпці можуть бути розташовані горизонтально або вертикально.

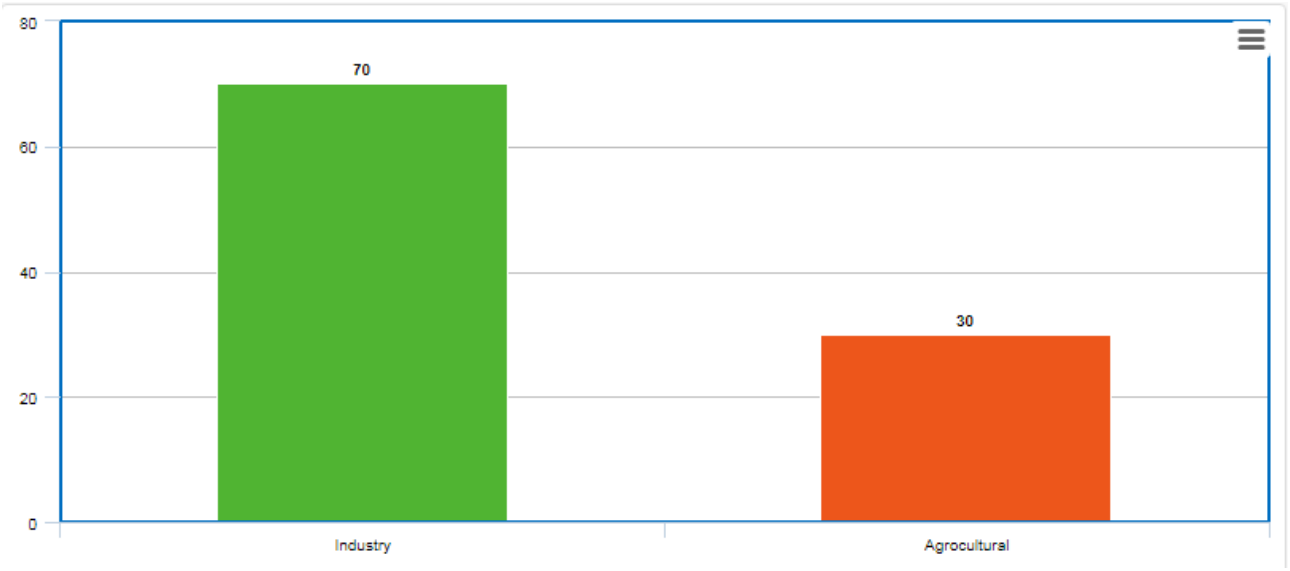


Рисунок 2.3 — Приклад стовпчикової діаграми

В порівнянні із секторними діаграмами, стовпчикові більш якісно візуалізують інформацію щодо пропорційного відношення для великої кількості складових, однак займають набагато більше місця.

Велику роль у графічному поданні відіграє вибір масштабу, який зробить його сприйняття кінцевими користувачами найбільш ефективним та сприятиме формулюванню коректних висновків про характер співвідношень в умовах предметної області.

Можна отримати висновок, що особливості використання стовпчикових діаграм дуже подібні до особливостей секторних, які найчастіше відображають кінцеві результати статистичних, економічних або наукових досліджень.

Однак секторні діаграми дозволяють отримати більшу компактність, при цьому жертвуючи потужністю, оскільки нехтують незначними складниками.

2.1.4 Подання даних на мапі

Візуалізація даних, що мають специфічний розподіл, найбільш ефективно проводиться з одночасною візуалізацією поверхні розподілу.

Прикладом такого розподілу є географічний.

В умовах географічного розподілу модель візуалізації повинна містити дві основні складові, а саме мапу, на якій зафіксовано розподіл, та власне дані, які міститимуть також інформацію для закріплення на мапі у вигляді координат, конкретних географічних місць тощо (рисунок 2.4).

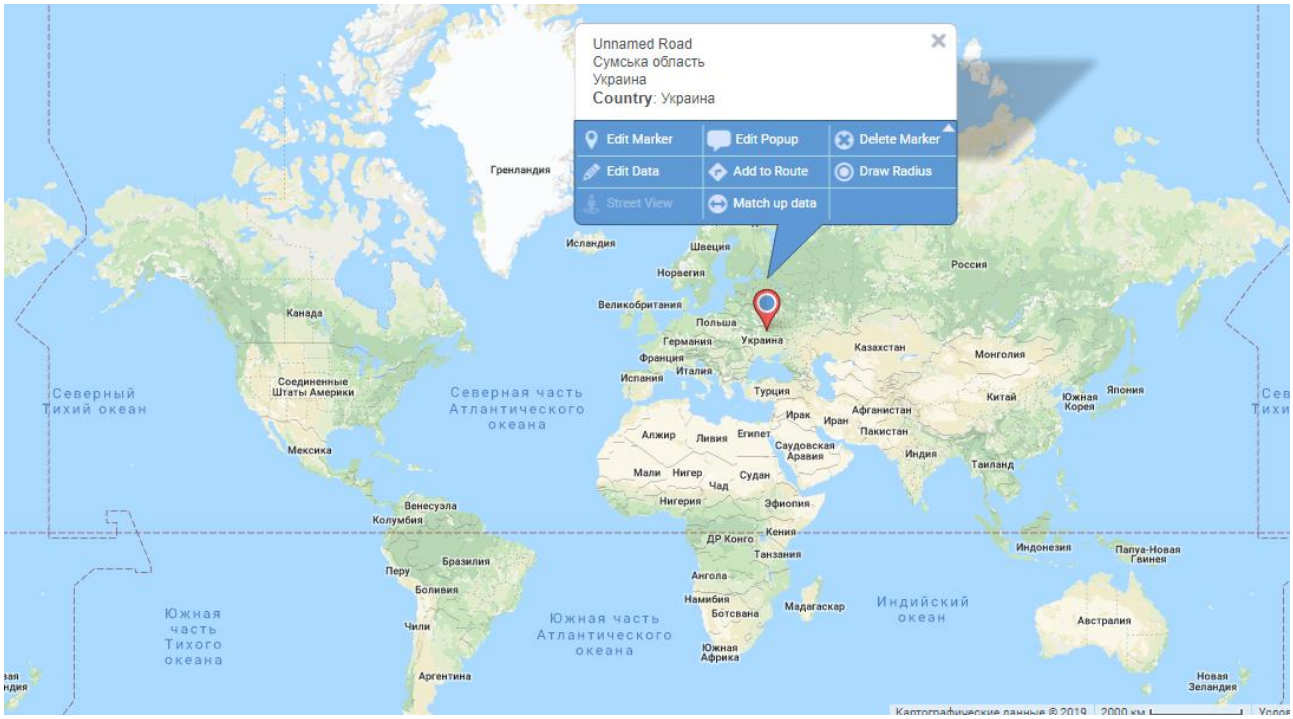


Рисунок 2.4 — Приклад розміщення даних на мапі

Такий розподіл надає потужну наочність, якщо обробляються географічно розподілені дані.

Зазвичай використовується мапа світу, яку можна масштабувати та перемикати між картографічним і супутниковим режимами тощо.

Однак більша частина даних, що потребує візуалізації, не має географічний розподіл та описує відносно складні системи.

2.1.5 Діаграма потоків

Діаграми потоків є прикладом складних технологій візуалізації.

Цей тип діаграм створений для відображення стану або потоку динамічних (тобто таких, що постійно змінюються) відношень у складній (такій, що містить багато неоднорідних компонентів) системі.

Метою побудови таких діаграм є підкреслення особливостей структури та взаємодії елементів різноманітних систем. Цілий комплекс результатів науково-прикладних досліджень може бути зображений у вигляді однієї такої діаграми.

Існує багато типів діаграм потоків, зокрема

— алювіальні,

— кругового потоку доходу,

— потоку контролю,

— кумулятивного потоку,

— блоку функціонального потоку,

— потоку даних,

— інформаційного потоку,

— станів.

Отже, залежно від того, з якого боку аналізувати взаємодію компонентів системи, можна побудувати структурно відмінні її графічні репрезентації.

Зокрема, алювіальні діаграми потоків відображають зміни у структурі мережі відносно часу (рисунок 2.4).

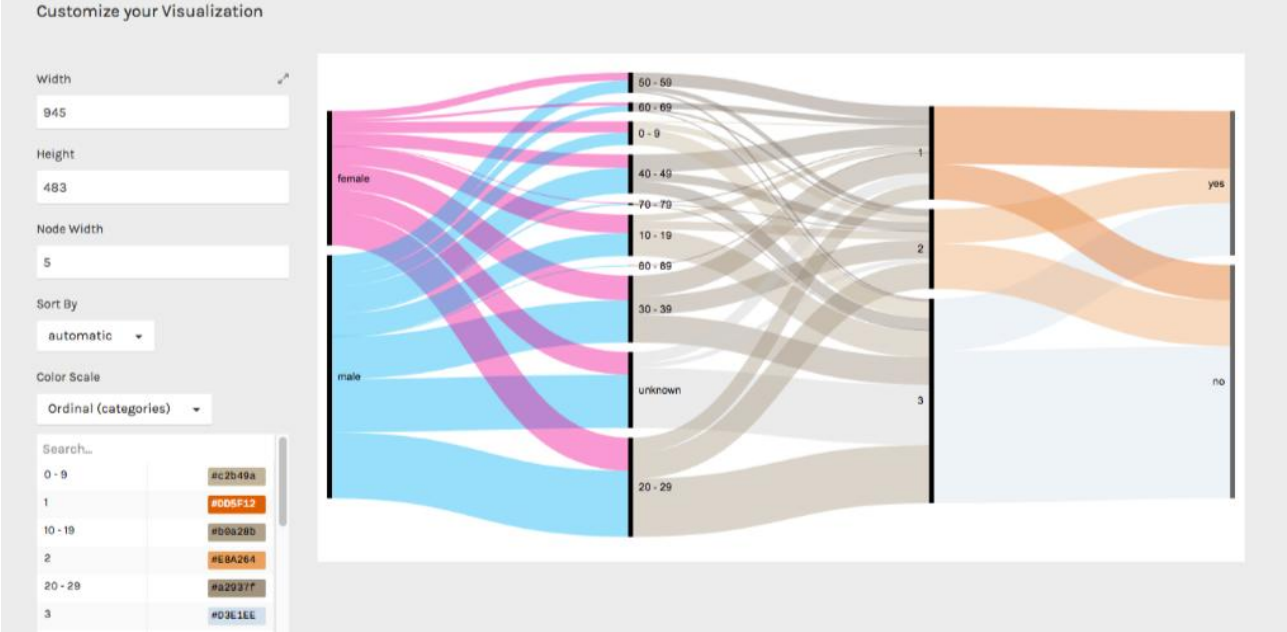


Рисунок 2.4 — Приклад алювіальної діаграми потоків для відображення залежності між статтю та віком пасажирів судна “Титанік” та їх загибеллю під час катастрофи[1]

2.2 Формати подання даних

Задача візуалізації та обміну даних полягає також у роботі з певними внутрішніми форматами їх представлення, оскільки у сучасному світі, пристосованому до ефективного опрацювання інформації, структура подання цієї інформації відіграє дуже велику роль.

Кожен алгоритм обробки даних спирається на формат, у якому ці дані подаються на вхід. До найпопулярніших форматів представлення даних традиційно відносять

- JSON,
- Excel,
- CSV.

Існує багато інших форматів, однак вони не мають такої популярності як зазначені або призначені для передачі та збереження інформації особливої структури.

2.2.1 Формат JSON

Спосіб нотації JSON (JavaScript Object Notation) є текстовим форматом для обміну даними, що спирається на синтаксис мови програмування JavaScript, однак сприймається засобами інших мов та є де-факто найпоширенішим стандартом для обміну даними у мережі Інтернет.

Популярність даного формату викликана його простотою, оскільки дані легко аналізуються як людьми, так і програмами. Фактично нотація є компромісом між чітким формалізмом, який потребують обчислювальні машини, та наочністю і легкістю побудови представлення, що потребують розробники та фахівці предметної області.

2.2.2 Формат Excel

Засіб Microsoft Excel використовується для побудови як простих, так і достатньо складних таблиць різної структури.

Таблиці, створені у програмі Excel, мають специфічне розширення та часто передаються через мережу Інтернет. Дані такого формату не записуються у формі простої колекції, як це відбувається у стандарті JSON, а мають краще підготовлену до візуалізації табличну форму.

2.2.3 Формат CSV

Стандарт CSV (coma separated value) подає дані як набір рядків, елементи яких розділені комами або іншими подібними роздільниками.

Як і формат Excel, CSV краще адаптований для попередньої підготовки даних до подальшої обробки, наприклад візуалізації, оскільки дані представлені у вигляді двомірної структури, на відміну від JSON, де вони є лінійним списком.

Програмний продукт Microsoft Excel має підтримку і для CSV-файлів, тобто не лише для інформації власного формату.

2.3 Аналіз існуючих програмних засобів

Існує низка програмних засобів для візуалізації числових даних у різних форматах:

— “Tableau” — це комплексна комп’ютерна система для управління та створення різноманітних діаграм та графіків;

— “VIGraph” — забезпечує повний контроль візуалізації, з основною метою забезпечення простоти у використанні для початківців і професійних користувачів.

Головним недоліком кожної із систем є те, що вони потребують спеціального програмного забезпечення й працювати з ними можливо тільки на спеціально обладнаному для цього комп’ютері.

Це викликає великі незручності, тому що потребує затрат як фінансових, так і часових.

Наступним недоліком “Tableau” є цінова політика для обміну даними між користувачами. Засіб “VIGraph” взагалі не має можливості обміну графіками між користувачами системи. Для початківців ці програми можуть здатися переповненими функціоналом та дуже специфічними в плані налаштування.

Також до недоліків цих систем можна віднести важкий для розуміння інтерфейс користувача, для повноцінного використання якого користувач повинен мати великий досвід роботи з комп’ютером та програмними системами зокрема. Зрозуміло, що користуватись даною системою зможуть далеко не всі потенційні користувачі.

Так як використання веб-додатків набуває широкого розповсюдження, то розробка зручної, інтуїтивно зрозумілої, гнучкої та невимогливої до апаратного забезпечення системи візуалізації числових даних та обміну між користувачами є актуальною задачею.

Одними із варіантів, який зможе працювати на будь-яких персональних комп’ютерів та на будь-якій операційній системі, є розробка web-додатку для цієї

системи. Кожен користувач зможе отримати доступ до свого власного проекту з будь-якого комп'ютера через браузер. Дана система не має прив'язки до конкретного пристрою, тому це стає дуже великою перевагою над системами, які потребують встановлення додаткового програмного забезпечення.

Як приклад можна навести web-систему "Tableau" з більш детальним описом та наведенням прикладів інтерфейсу програми.

Основними можливостями програми "Tableau" є:

— завантаження файлів у форматах excel, csv та json;

— вибір режиму перегляду за допомогою меню, яке випадає;

— вибір форматування даних;

— налаштування діаграм під кожний набір даних;

— можливість змінювати конфігурацію вже побудованої діаграми;

— вивантаження інформації та діаграми;

— зберігання побудованої діаграми для подальшого використання та робити з нею;

— можливість корегування налаштувань проектів, які вже були завантажені;

— автоматичне відображення детальної інформації про кожну діаграму.

На рисунках 2.5 — 2.6 наведені приклади роботи даної системи, які демонструють особливості зовнішнього вигляду системи та реалізацію основних функцій.

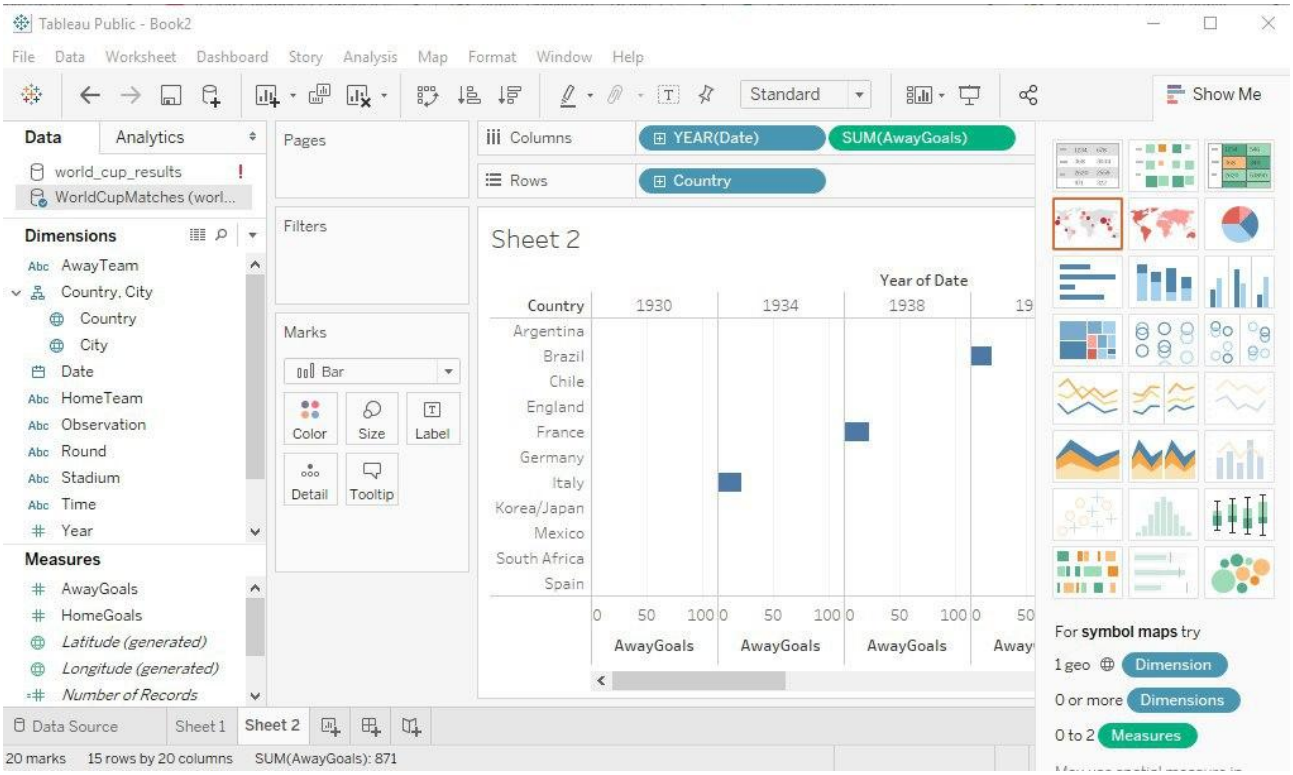


Рисунок 2.5 — Інтерфейс програми Tableau для роботи з діаграмами

Як видно із зображень, інтерфейс є достатньо складним для швидкого освоєння новими користувачами, що не мають досвід роботи з подібним програмним забезпеченням. Необхідно вивчити англомовну термінологію для предметної області інструментарію візуалізації, що потребує додаткового часу та зусиль від потенційних користувачів програмного засобу.

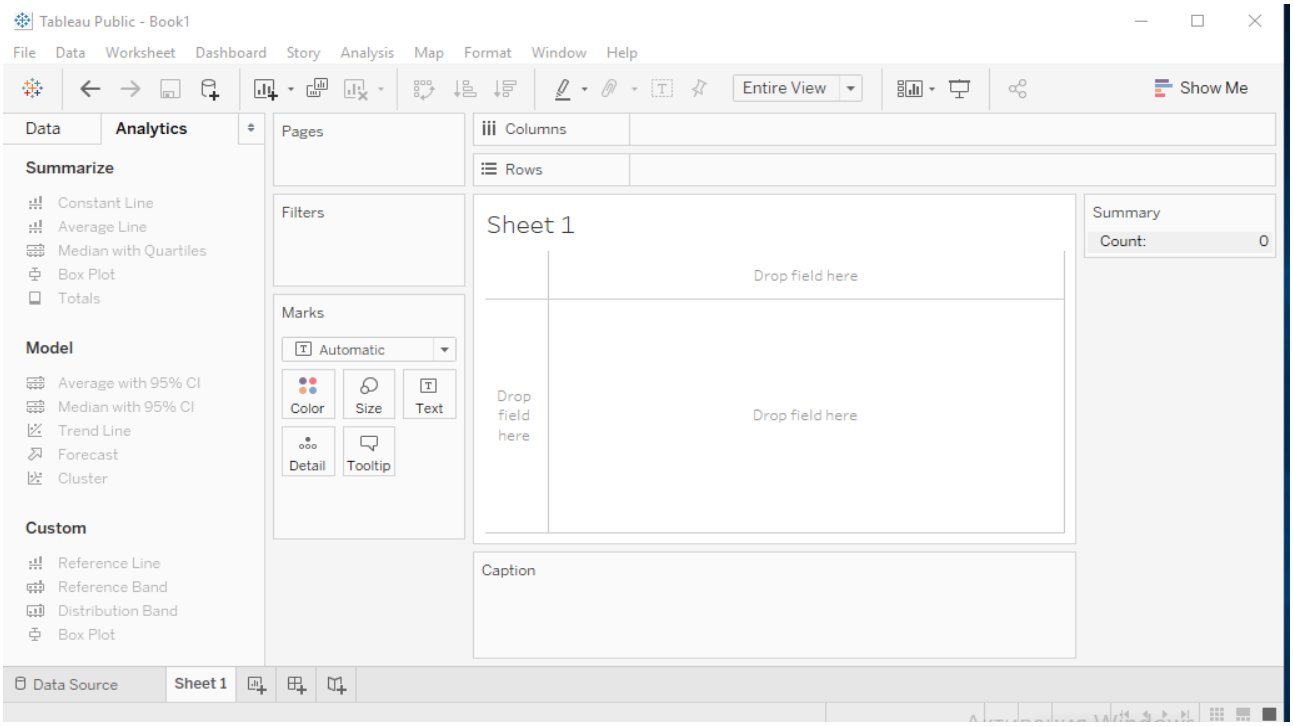


Рисунок 2.6 — Відображення головного меню Tableau

2.4 Висновки до розділу

У даному розділі було описано особливості форматів графічного представлення числових даних та існуючі програмні рішення, що дозволяють виконувати побудову зручних для сприйняття кінцевими користувачами графіків та діаграм.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Відповідно до поставленої задачі було обрано сучасні та гнучкі засоби та середовища розробки програмного забезпечення.

Вибір конкретного засобу залежить від багатьох факторів, зокрема

— обраної архітектурної концепції;

— переваг та недоліків засобу над аналогічними рішеннями;

— наявності якісної документації;

— його популярності (що означає наявність спілки активних користувачів)

3.1 Вибір концепції системи

Шаблон проектування — це рішення, яке описує яким чином вирішуються задачі, які часто зустрічаються при розробці програмних систем [3].

Підхід MVP (“Model-View-Presenter”) — це шаблон проектування, похідний від MVC (“Model-View-Controller”), головною ідеєю якого є відділення логіки застосунку від представлення (рисунок 3.1). Даний архітектурний шаблон використовується для побудови зручних та потужних користувацьких інтерфейсів. Принципом MVP є розділення програмної реалізації системи на три головні компоненти: M — Model (Модель), V — View (Представлення), P — Presenter (Представник). Розділення відбувається таким чином, щоб редагування будь-якого компонента могло відбуватися незалежно [3].

Модель містить знання про предметну область, дані та правила у чіткому форматі, але не має інформації про контролери та представлення. У моделі реалізується бізнес-логіка роботи застосунку. Модель надає контролеру дані, які запитує користувач або інша система.

Представлення надає можливість по-різному відображати дані, отримані будь-яким способом від моделі. Цей шар може містити в собі логіку, але тільки таку, що

стосується механізму візуалізації кінцевих даних. Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Представник реалізує взаємодію між моделлю і представленням.

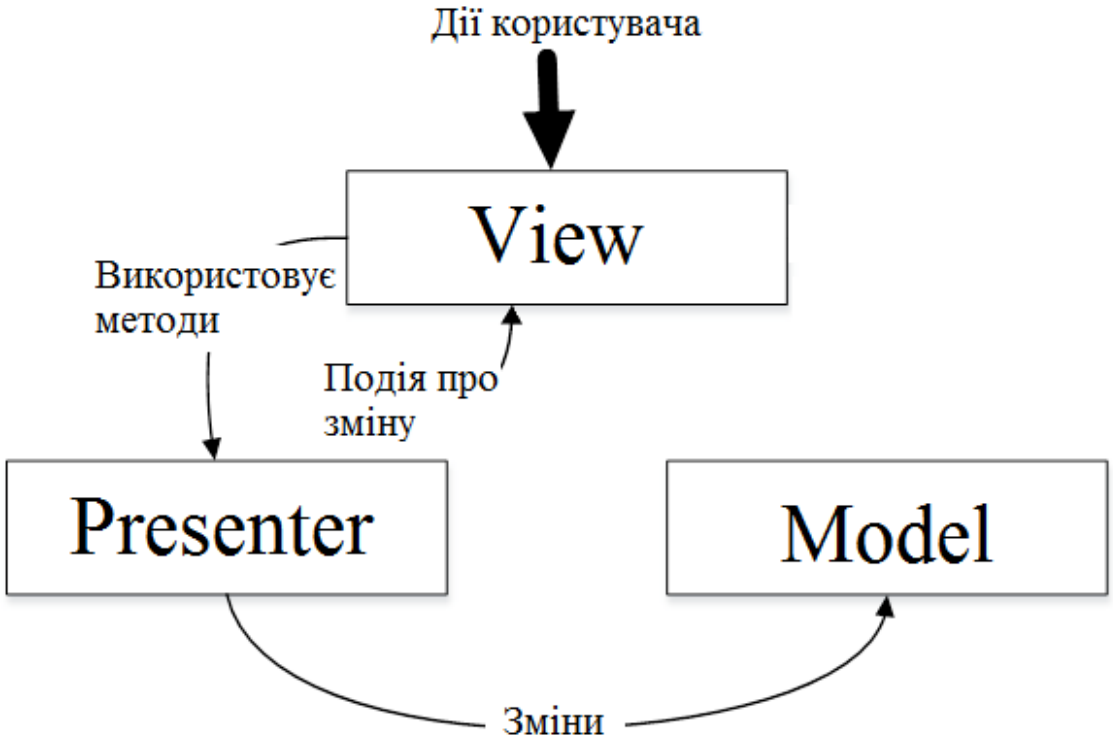


Рисунок 3.1 – Схема роботи MVP шаблону

Комунікація між окремими складовими досягається використанням механізму подій та фіксації змін.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу реалізується можливість повторного використання окремих компонентів програмного комплексу та гнучкість всієї системи.

3.2 Опис інструментів розробки

Для написання коду програмного продукту було використано мову програмування JavaScript.

Подібний вібір обумовлений зручністю мови JavaScript для створення веб-продуктів, в тому числі їх серверних компонентів.

Збереження даних відбувається за допомогою системи управління базами даних PostgreSQL.

Клієнтська сторона написана за допомогою фреймворку Angular, розробленого компанією Google. Angular використовує TypeScript (типізоване розширення мови JavaScript) та дозволяє ефективно контролювати масштабованість системи.

3.2.1 Angular

Фреймворк Angular (рисунок 3.2) було обрано для реалізації клієнтської частини системи, оскільки він підтримує велику кількість ефективних методологічних підходів до створення потужного інтерфейсу користувача та надає можливість як якісно організовувати програмний код відповідно до компонентної структури застосунку, так і зручні для розробника точки інтеграції системи із зовнішніми бібліотеками.

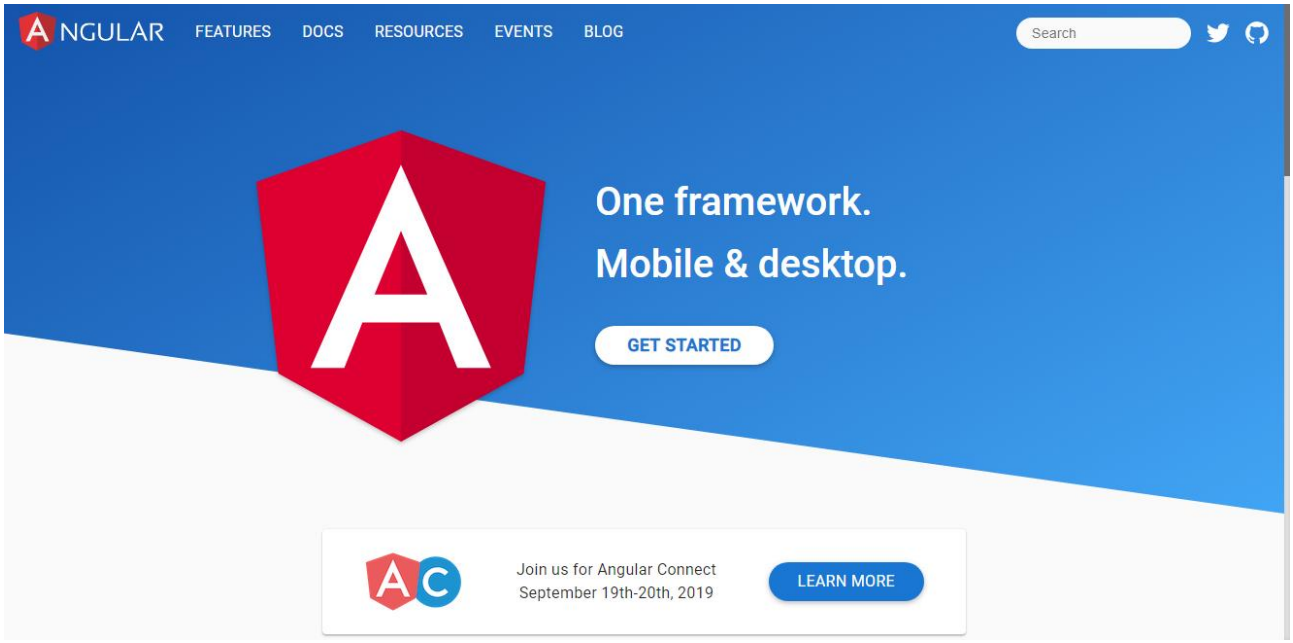


Рисунок 3.2 – Головна сторінка проекту Angular

Система, написана з використанням даного фреймворку, містить сервіси для обробки даних, що надходять із серверу, та компоненти, які використовують методи сервісів для відображення даних та відповіді на запити кінцевих користувачів.

Згідно принципів ефективного веб-дизайну зовнішній інтерфейс системи представляється у вигляді компонентів, які можуть містити інші компоненти або бути остаточними у ланці представлення.

Програмний код кожного компоненту містить чотири файли:

- шаблон сторінки формату HTML,
- стилі сторінки формату CSS,
- код компоненту на мові TypeScript,
- модульні тести компоненту на мові TypeScript.

Подібні файли компонент зберігаються разом, але можуть редагуватися та тестуватися окремо, що є перевагою при розробці складних систем.

Фреймворк Angular було обрано для програмної реалізації системи, оскільки він дуже ефективно імплементує принципи компонентного програмування, дозволяючи писати менше програмного коду для досягнення більшої ефективності та редагувати кожний компонент як незалежну програму.

3.2.2 Бібліотека NgRx

Було обрано бібліотеку NgRx для створення єдиного стану системи та спрощення маніпуляції з даними. Єдиний стан для системи дозволяє відокремити логіку збереження інформації від її обробки та представлення у компонентах. Рішення NgRx дозволяє створювати реактивні застосунку на основі фреймворку Angular. Під реактивністю розуміється ефективний асинхронний відклик програмної системи на будь-які дії користувача, що викликають події інтерфейсу.

Бібліотека NgRx забезпечує управління станом, ізоляцію побічних ефектів, управління колекціями об'єктів, прив'язку маршрутизатора, генерацію коду та інструменти розробника, які підвищують досвід розробників при побудові багатьох різних типів програм.

Основні принципи, що постулюються розробниками цього рішення:

- стан є єдиною, незмінною структурою даних;
- компоненти делегують відповідальність за побічні ефекти, які обробляються в ізоляції;
- типова безпека просувається в архітектурі з урахуванням компілятора TypeScript для загальної синтаксичної коректності програми;
- дії та стан можуть бути серіалізовані, щоб гарантувати, що стан передбачувано зберігається і відтворюється;
- застосування цієї бібліотеки сприяє використанню парадигми функціонального програмування при побудові реактивних додатків;
- забезпечення простої стратегії тестування для перевірки функціональності.

Отже, рішення NgRx було обрано для використання у програмній системі, оскільки воно є природньою складовою системи Angular та надає відчутні переваги в управлінні даними, які мають бути доступні одночасно у декількох модулях системи.

3.2.3 Бібліотека D3

Візуалізація даних є найважливішою та найскладнішою частиною системи, що розроблюється. Для реалізації цієї непростої задачі було прийнято рішення обрати низькорівневу бібліотеку, оскільки це дозволяє отримати максимально повний контроль над процесом виконання програмного комплексу.

Було обрано рішення D3 (Data-Driven Documents), яке надає змогу будувати діаграми різних рівнів складності. Бібліотека D3 дозволяє прив'язувати довільні дані до об'єктної моделі документа (DOM), а потім застосовувати до цілого документу перетворення, керовані даними. Наприклад, існує можливість використання D3 для створення HTML-таблиці з масиву чисел. Іншим прикладом є створення інтерактивної гістограми формату SVG з плавними переходами між окремими компонентами зображення. Взагалі існує велика кількість можливих прецедентів застосування даного рішення, багато з яких показано на головній сторінці проекту (рисунок 3.3).

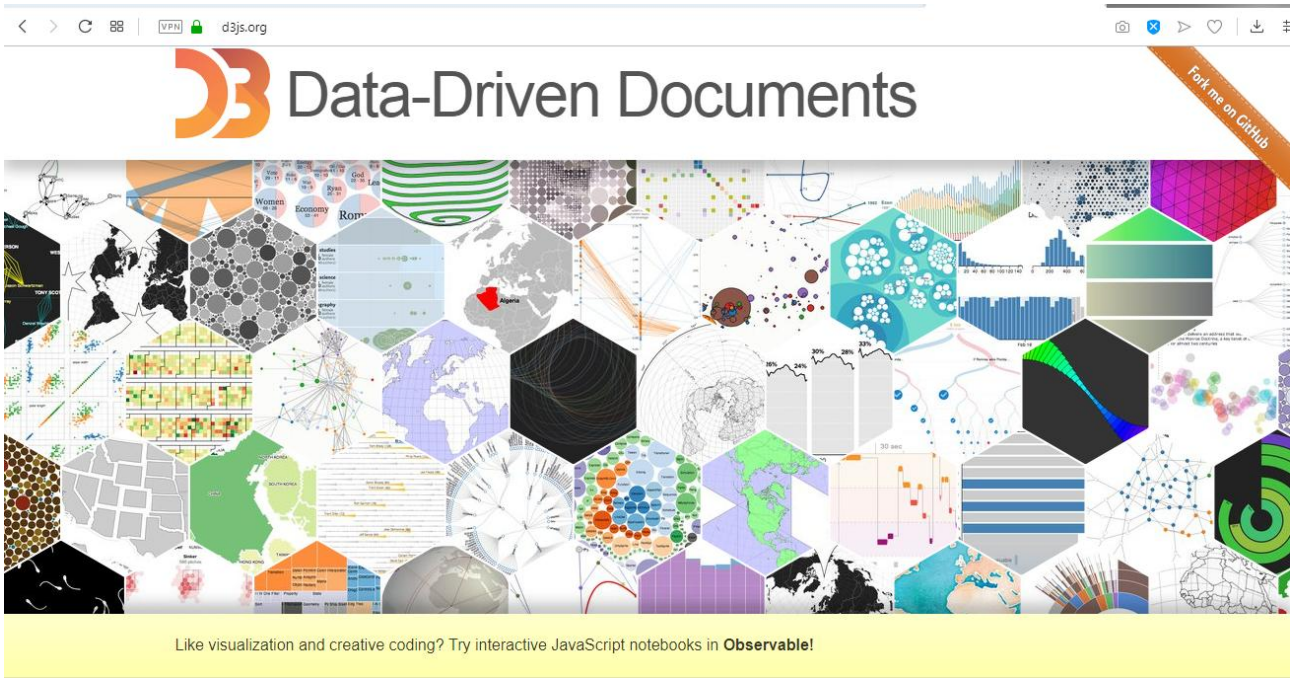


Рисунок 3.3 – Головна сторінка проекту Data-Driven Documents з прикладами застосування

Рішення D3 не є монолітною бібліотекою, яка прагне забезпечити кожну теоретично можливу функцію. Натомість засоби D3 вирішують суть проблеми: ефективне маніпулювання документами на основі даних. Це дозволяє уникнути хибних власних уявлень розробника і надає надзвичайну гнучкість, розкриваючи всі можливості таких веб-стандартів, як HTML, SVG і CSS. З достатньо великою ефективністю D3 підтримує обробку відносно великих наборів даних і динамічну поведінку для взаємодії та анімації. Функціональний стиль D3 дозволяє повторно використовувати написаний програмний код через різноманітний набір офіційних модулів, розроблених спільнотою активних користувачів даного рішення.

3.2.4 Платформа Node.js

Програмний засіб Node.js — серверна платформа для асинхронного JavaScript–оточення, робота якого заснована на подіях.

Засіб Node спроектований для побудови масштабованих мережевих додатків. Платформа була створена під впливом таких систем як Event Machine в Ruby і Twisted в Python.

Програма Node використовує подієву модель значно ширше, ніж аналоги, оскільки цикл подій (event loop) було прийнято за основу оточення замість того, щоб використовувати його в якості бібліотеки. В інших подібних системах завжди виконується блокування виклику, щоб запустити цикл подій. Зазвичай поведінка визначається через функції зворотнього виклику на початку програмного коду, а в кінці запускається сервер через блокуючий (синхронний) виклик як EventMachine::run(). В Node немає нічого подібного до виклику початку циклу подій. Відбувається просте входження до подієвого циклу після запуску скрипта на виконання. Отже, система Node виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотнього виклику. Така поведінка схожа на поведінку браузерного JavaScript: подієвий цикл прихований від користувача.

3.2.5 Система управління базами даних PostgreSQL

Засіб PostgreSQL (рисунок 3.4) є надзвичайно багатий функціональними можливостями, містить велику кількість засобів налаштування і способів їх реалізації та розширення для задоволення потреб конкретного розробника.

Даний засіб управління сховищем даних вважається дуже надійним та часто використовується для забезпечення роботи багатьох систем, в тому числі великих підприємств. Однак ефективність рішення не зменшується при його використанні у невеликих системах.

Система має відкритий код, потужну систему управління даними, реалізує стандарт мови SQL (Structured Query Language), в тому числі програмний доступ до баз даних.

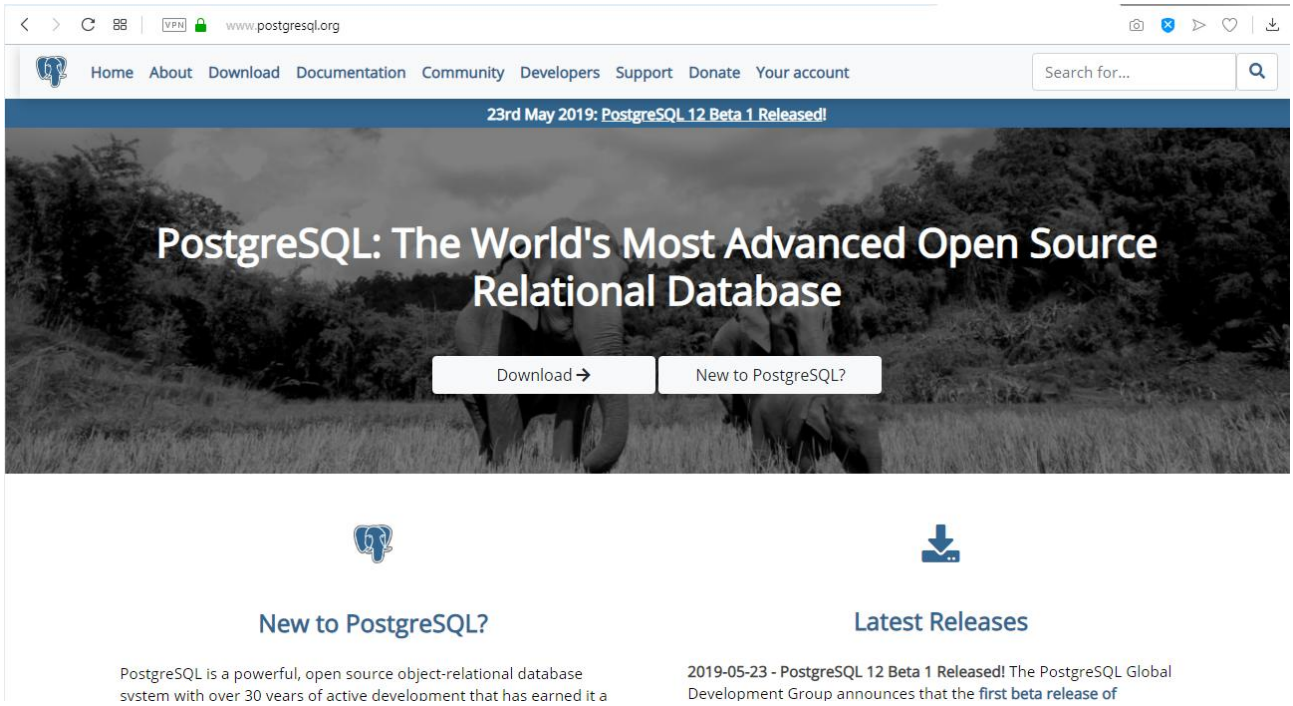


Рисунок 3.4 – Головна сторінка проекту PostgreSQL

Було використано модуль pg для роботи з базою даних PostgreSQL. Це просте та ефективне рішення для проектів на мові JavaScript.

3.2.6 Фреймворк Jest

При розробці систем будь-яких масштабу важливу роль відіграє підхід програмування, що спирається на результати тестів (test-driven development).

Використання такого підходу дозволяє швидко знаходити та виправляти помилки в роботі окремих компонентів системи, чітко уявляти їх призначення та очікувану поведінку.

Для реалізації цього підходу було обрано рішення Jest (рисунок 3.5).

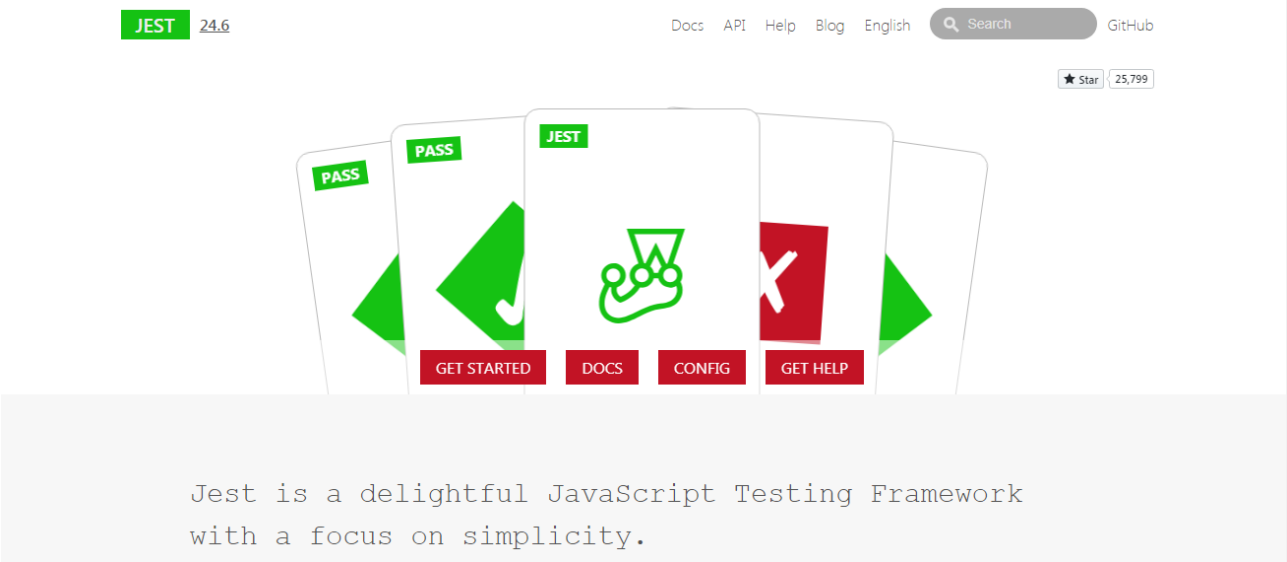


Рисунок 3.5 – Головна сторінка проекту Jest

Фреймворк Jest має виразний акцент на простоту. Розробникам, що використовують дане рішення, потрібно виконувати лише мінімальну конфігурацію. Підтримується механізм паралельного виконання тестів для досягнення максимально можливої ефективності. Система Jest особливо обробляє тести, що невдало завершилися раніше, що робить її унікальним рішенням у порівнянні з аналогами.

3.3 Опис середовища розробки Visual Studio Code

Для безпосереднього написання програмного коду систему та налаштування її глобального стану було обрано середовище розробки Visual Studio Code.

Даний текстовий редактор, створений Microsoft, відрізняється можливістю встановлення різноманітних плагінів (додатків), створених спільнотою користувачів, що робить його легким та потужним середовищем розробки, яке можна налаштувати для роботи з практично будь-якими засобами розробки, в тому числі JavaScript фреймворками (рисунок 3.6).

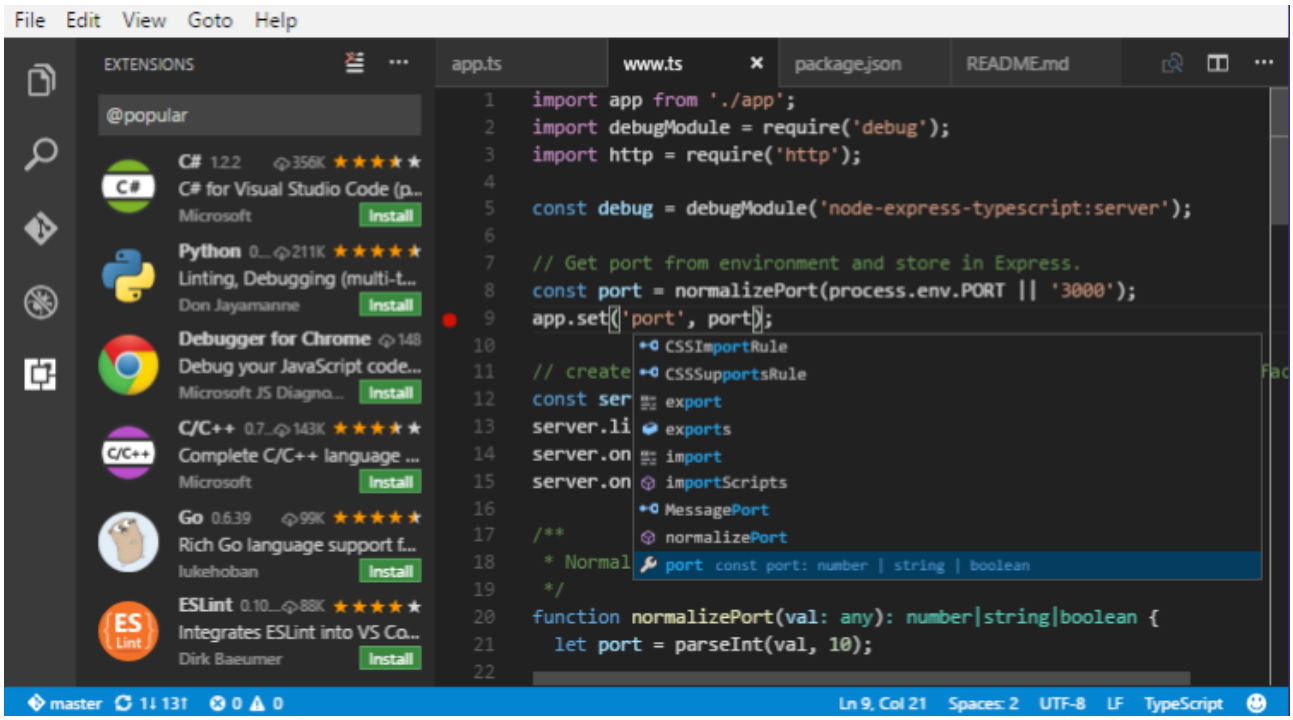


Рисунок 3.6 – Інтерфейс середовища Visual Studio Code

Налаштоване для веб-розробки середовище надає засоби підсвічення синтаксису, генерації та автодоповнення для мов TypeScript, JavaScript, HTML, CSS тощо.

Таким чином, використання всіх можливостей Visual Studio Code перетворює дане рішення з простого редактора для набору рядків коду у потужний засіб, що дозволяє писати програмний код швидше, автоматично виправляти помилки, що

виникають через неуважність тощо. Використання темної кольорової теми є ергономічним та зменшує загрозу очам розробника, який працює з екранним пристроєм значні обсяги часу.

3.4 Висновки до розділу

Було описано основні шаблони проектування, програмні засоби розробки, середовище розробки, обрані для створення програмного коду системи, зазначено їх переваги над аналогами.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Створена система складається з шести основних логічних модулів.

Модуль авторизації відповідає за реєстрацію нових користувачів у системі та авторизацію існуючих.

Модуль обробки даних обслуговує процес завантаження користувацьких даних до системи. При цьому підтримуються різні формати представлення даних.

Модуль вибору певного типу графіку або діаграми може бути легко розширений та задає основні правила побудови конкретної графічної візуалізації.

Модуль відображення та конфігурації певного типу графіка під вимоги користувача надає користувачу засоби та можливості управління структурою та графічними характеристиками візуалізації.

Модуль експорту створеної візуалізації надає функціонал для завантаження діаграми або її інтеграції у довільну веб-сторінку користувача.

Модуль управління проектами відображає у вигляді списку всі проекти системи, до яких конкретний користувач має доступ. За допомогою цього модулю реалізовано механізм надання спільного доступу іншим користувачам до власних проектів.

Така структура реалізує основні принципи компонентного програмування, оскільки кожен з модулів може розглядатися як відносно незалежна програма, та відображає сценарій дій користувач, являючи собою логічну модель використання програмного комплексу.

На рисунку 4.1 наведені відношення залежності між окремими модулями системи. Модуль авторизації надає доступ авторизованим користувачам до управління існуючими проектами та створення нових.

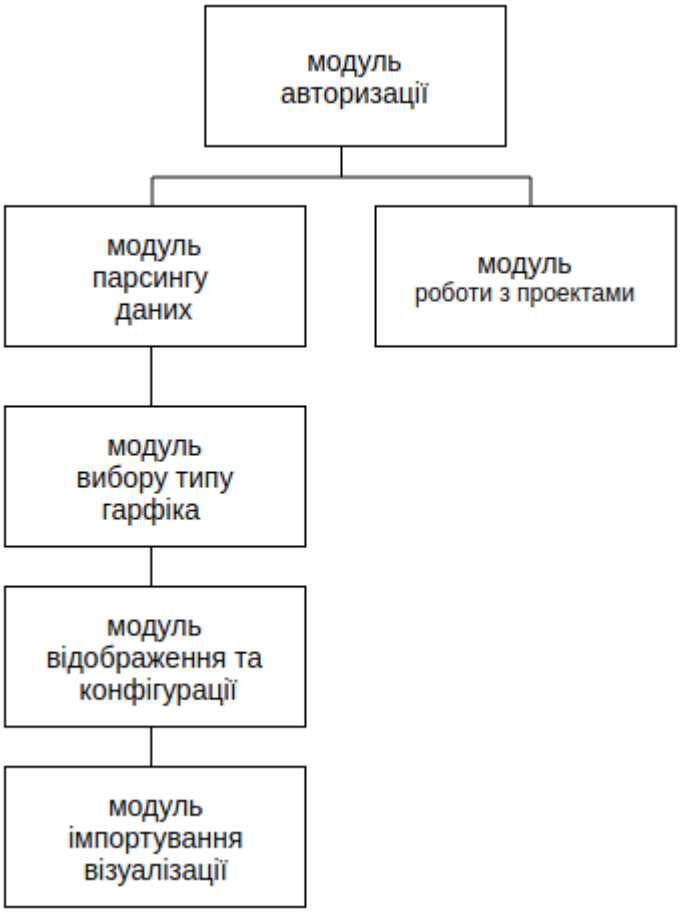


Рисунок 4.1 — Логічна структура веб-застосунку

З точки зору архітектури система реалізує класичний для веб-систем триланковий підхід (рисунок 4.2).

Рисунок 4.2 — Загальна архітектура програмної системи та комунікація між її основними складовими

Клієнт взаємодіє із системою за допомогою веб-браузера, який отримає на вхід кінцеві дані від клієнтської частини та дозволяє користувачу викликати події, що призводять до зміну стану системи.

У клієнтській частині реалізовано логіку представлення.

Клієнтська частина посилає HTTP-запити до серверної, коли потребує використання методів бізнес-логіки.

Серверна частина реалізує алгоритми візуалізації, обробляючи дані від користувачів або з бази даних. Для забезпечення останнього вона має функціонал виконання SQL-запитів по відношенню до бази даних, що також використовується, якщо певна зміна стану системи повинна бути зафіксована як довгострокова.

4.1 Структура бази даних

Основу бази даних системи (рисунок 4.3) складають відношення між користувачами та проектами, при цьому користувачі розподілені за групами та компаніями.

Варто зазначити, що між сутностями користувачів та проектів існує співвідношення “один до багатьох”, а не “багатьох до багатьох”, оскільки у кожного проекту є тільки один власник, що, втім, не протидіє можливості його спільного використання іншими користувачами за дозволу власника.

Відомості про користувача містять його електронну адресу, оскільки вона використовується для надання спільного доступу замість імені користувача.

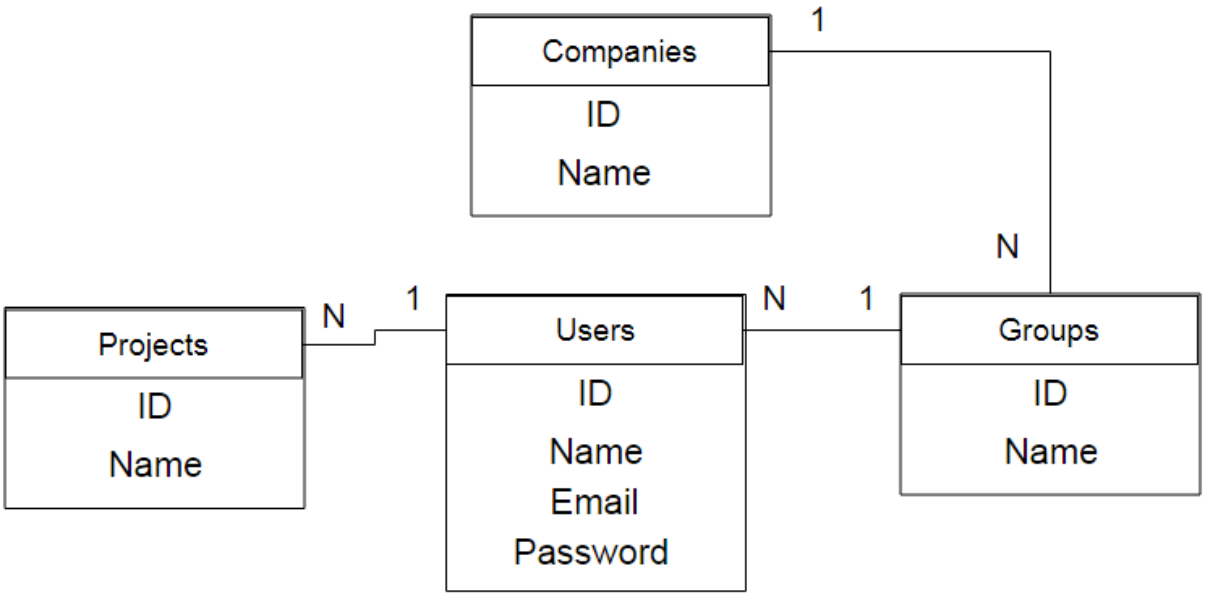


Рисунок 4.3 — Логічна схема основних компонентів бази даних системи

Схема даних системи є надзвичайно складною та містить інші сутності, які відповідають за збереження у системі даних щодо створених діаграм та можливостей їх модифікації.

4.2 Структура серверної частини

Серверна частина веб-системи є шаром бізнес-логіки, що покликаний виконувати декілька незалежних завдань:

— аутентифікацію та авторизацію користувачів;

— збереження наборів даних у базі;

— генерації діаграми у форматі HTML;

— завантаження шаблонів графіків;

— регулювання доступу до проектів.

4.3 Структура клієнтської частини

Клієнтська частина веб-застосунку була розділена на компоненти-модулі відповідно до логічної структури системи.

До цих модулів належать наступні:

— модуль авторизації;

— модуль управління проектами;

— модуль обробки даних;

— модуль вибору типу графіка;

— модуль відображення та конфігурації;

— модуль експорту візуалізації.

Кожен модуль має свої чітко визначені функції.

4.3.1 Модуль авторизації

Даний модуль дозволяє користувачу створити обліковий запис або увійти до системи за умови існування останнього.

Всі функції компонента стосуються управління обліковими записами користувачів:

— створення облікового запису;

— ідентифікація користувача;

— вхід в систему;

— валідація даних для входу чи реєстрації;

— відображення підказок при виникненні помилок.

4.3.2 Модуль обробки даних

Даний модуль дозволяє користувачу завантажити дані для подальшого використання в системі. Його функції включають:

— додавання даних через текстове поле;

— додавання даних через посилання;

— додавання даних через завантаження файлу;

— перегляд завантаженої інформації;

— видалення даних;

— зміни раніше введеної інформації.

4.3.3 Модуль вибору типу графіка

Даний модуль дозволяє користувачу обрати загальну структуру графіка, який буде у подальшому використовуватися для візуалізації його даних. Повний список функцій компоненту включає:

— перегляд інформації про тип графіку;

— вибір одного варіанту з можливих;

— можливість змінювати вже обраний тип графіку;

— відображення візуалізації для кожного з типів графіку.

4.3.4 Модуль відображення та конфігурації

Даний модуль дозволяє користувачу індивідуально налаштувати основні параметри графіку для своїх даних. Функції компоненту включають:

— вибір стовпців для графіку;

— вибір рядків для графіку;

— вибір розміру діаграми;

— налаштування відступів візуалізації;

— налаштування висоти;

4.3.5 Модуль експорту візуалізації

Даний модуль дозволяє користувачу отримати результати побудови графіку у потрібному форматі, реалізовуючи:

— копіювання отриманого графіку у форматі зображення;

— отримання представлення графіка для подальшого використання на веб-сторінці.

4.3.6 Модуль управління проектами

Даний модуль дозволяє користувачу працювати з проектами, а саме:

— видаляти проект;

— змінювати назву проекту;

— редагувати вміст проекту;

— надавати спільний доступ до проекту іншим користувачам.

4.4 Висновки до розділу

У даному розділі було описано детальну структуру та функції окремих модулів програмної системи.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблена програмна система працює у середовищі мережі Інтернет, що забезпечує можливість працювати з нею, використовуючи будь-які апаратні платформи та операційні системи, які підтримують веб-браузери.

5.1 Інсталяція та системні вимоги

У випадку відсутності засобів для підключення до мережі Інтернет та перегляду веб-сторінок їх необхідно встановити на обчислювальний пристрій користувача.

Для розгортки системи на сервері необхідно виконати команду “npm start”. Фізичний сервер для цього повинен відповідати мінімальним технічним вимогам:

— мінімум 2 гігабайти оперативної пам’яті;

— повинен бути встановлений фреймворк node.js, що управляє налаштуваннями програм на мові JavaScript;

— тактова частота процесору від 2 ГГц.

Для доступу до розгорнутої на сервері системи потрібно у веб-браузері перейти за адресою. Адреса визначається під час налаштування програмного комплексу на сервері.

5.2 Сценарій роботи користувача з системою

Для доступу до функціоналу системи новий користувач повинен зареєструватися (рисунок 5.1).

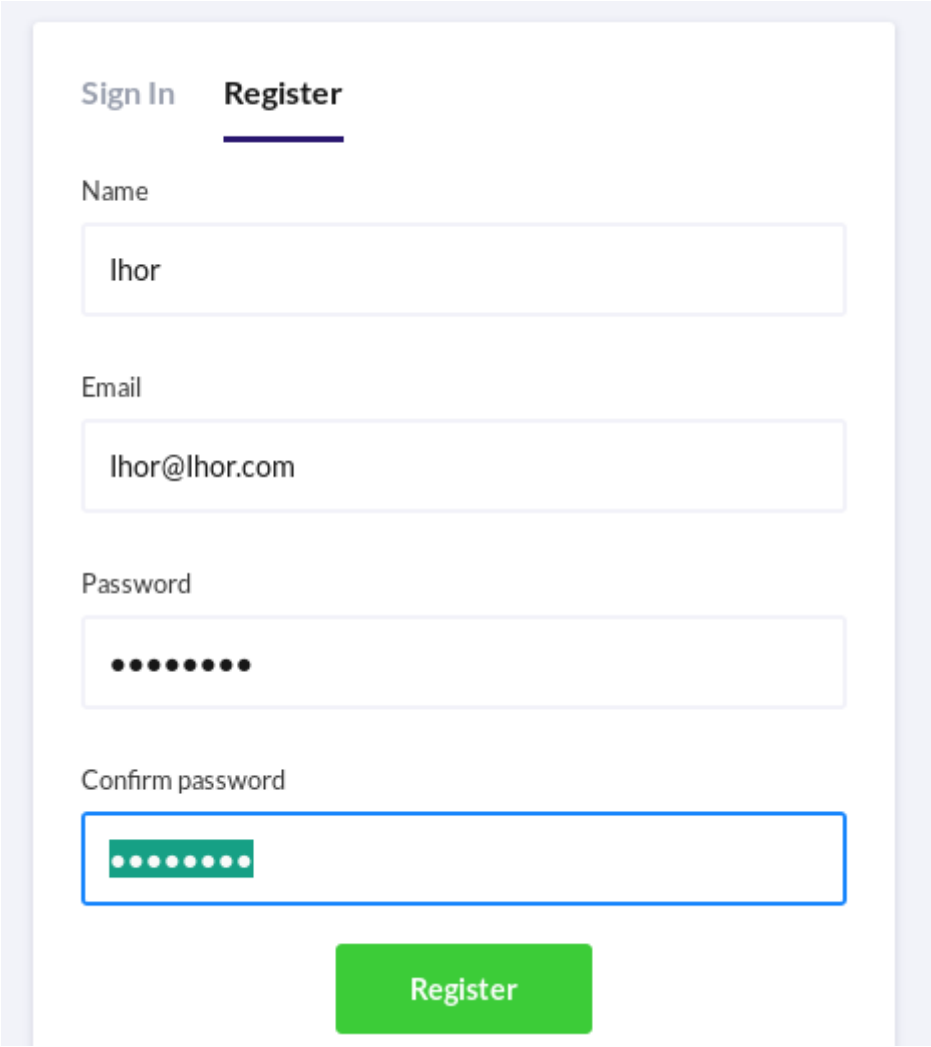
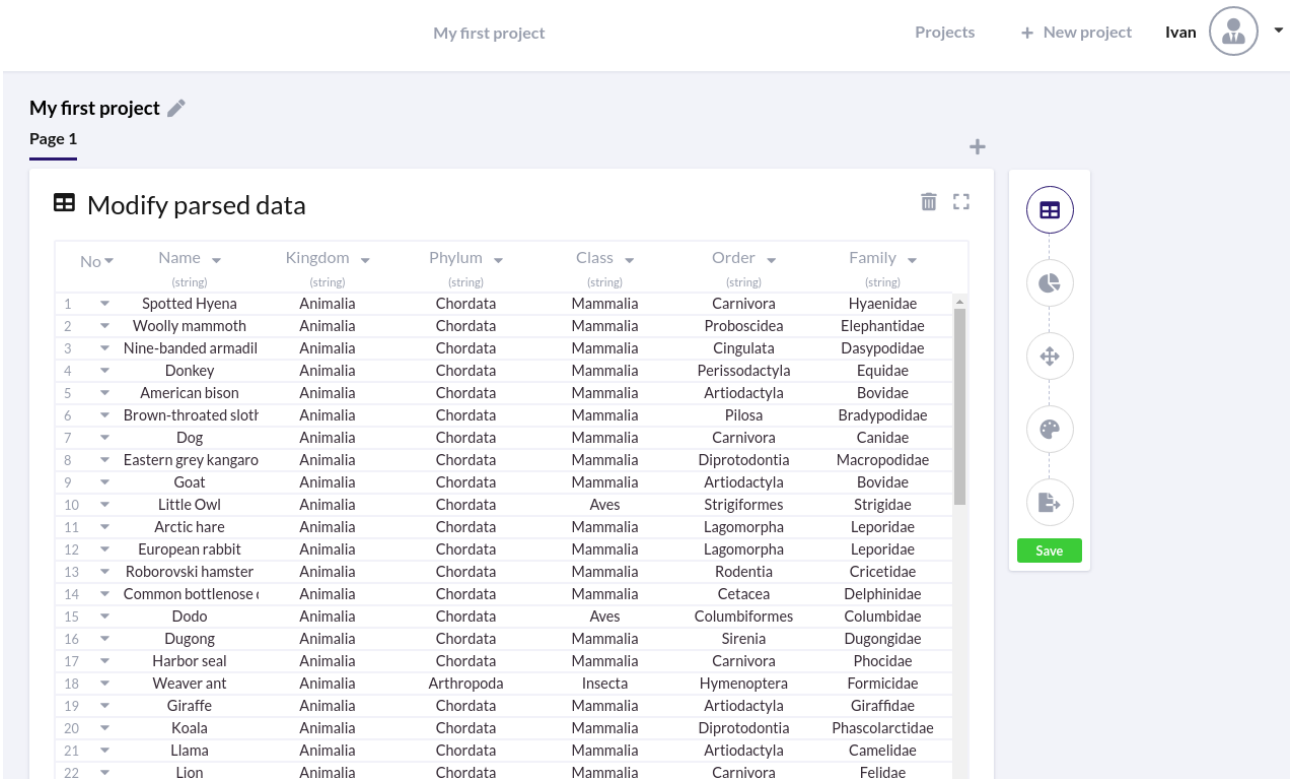


Рисунок 5.1 – Форма створення нового користувача

Після реєстрації користувач може створювати свої проекти. На першому етапі створення кожного проекту необхідно ввести дані для графічної візуалізації (рисунок 5.2).



No	Name (string)	Kingdom (string)	Phylum (string)	Class (string)	Order (string)	Family (string)
1	Spotted Hyena	Animalia	Chordata	Mammalia	Carnivora	Hyaenidae
2	Woolly mammoth	Animalia	Chordata	Mammalia	Proboscidea	Elephantidae
3	Nine-banded armadil	Animalia	Chordata	Mammalia	Cingulata	Dasypodidae
4	Donkey	Animalia	Chordata	Mammalia	Perissodactyla	Equidae
5	American bison	Animalia	Chordata	Mammalia	Artiodactyla	Bovidae
6	Brown-throated sloth	Animalia	Chordata	Mammalia	Pilosa	Bradypodidae
7	Dog	Animalia	Chordata	Mammalia	Carnivora	Canidae
8	Eastern grey kangaro	Animalia	Chordata	Mammalia	Diprotodontia	Macropodidae
9	Goat	Animalia	Chordata	Mammalia	Artiodactyla	Bovidae
10	Little Owl	Animalia	Chordata	Aves	Strigiformes	Strigidae
11	Arctic hare	Animalia	Chordata	Mammalia	Lagomorpha	Leporidae
12	European rabbit	Animalia	Chordata	Mammalia	Lagomorpha	Leporidae
13	Roborovski hamster	Animalia	Chordata	Mammalia	Rodentia	Cricetidae
14	Common bottlenose	Animalia	Chordata	Mammalia	Cetacea	Delphinidae
15	Dodo	Animalia	Chordata	Aves	Columbiformes	Columbidae
16	Dugong	Animalia	Chordata	Mammalia	Sirenia	Dugongidae
17	Harbor seal	Animalia	Chordata	Mammalia	Carnivora	Phocidae
18	Weaver ant	Animalia	Arthropoda	Insecta	Hymenoptera	Formicidae
19	Giraffe	Animalia	Chordata	Mammalia	Artiodactyla	Giraffidae
20	Koala	Animalia	Chordata	Mammalia	Diprotodontia	Phascolarctidae
21	Llama	Animalia	Chordata	Mammalia	Artiodactyla	Camelidae
22	Lion	Animalia	Chordata	Mammalia	Carnivora	Felidae

Рисунок 5.2 – Форма внесення даних для подальшої візуалізації

На другому етапі користувач має змогу обрати тип діаграми та відобразити структуру раніше введених ним даних у виміри графічного представлення (рисунок 5.3).

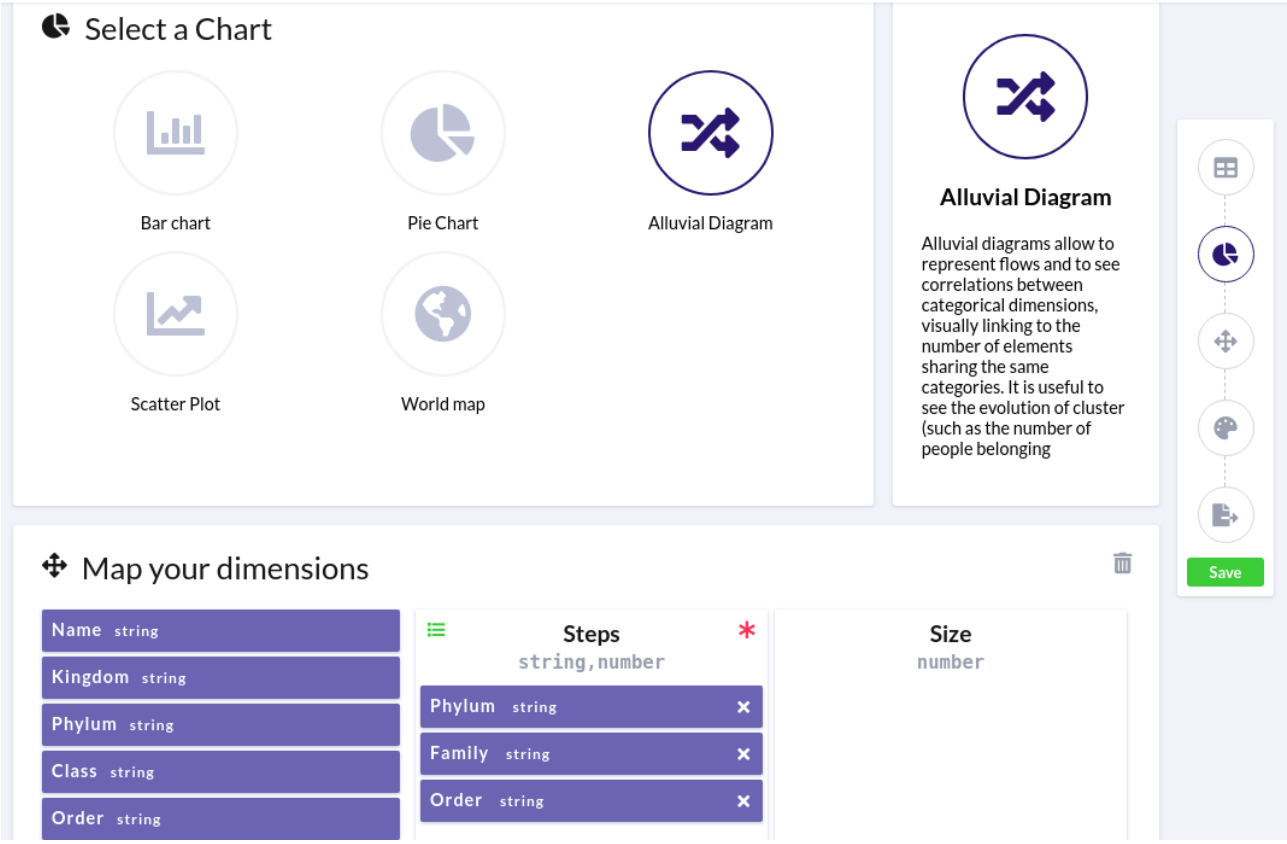


Рисунок 5.3 – Форма вибору типу та основних характеристик діаграми

На основі попереднього вибору користувача створюється ескіз діаграми, який можна відкорегувати (рисунок 5.4).

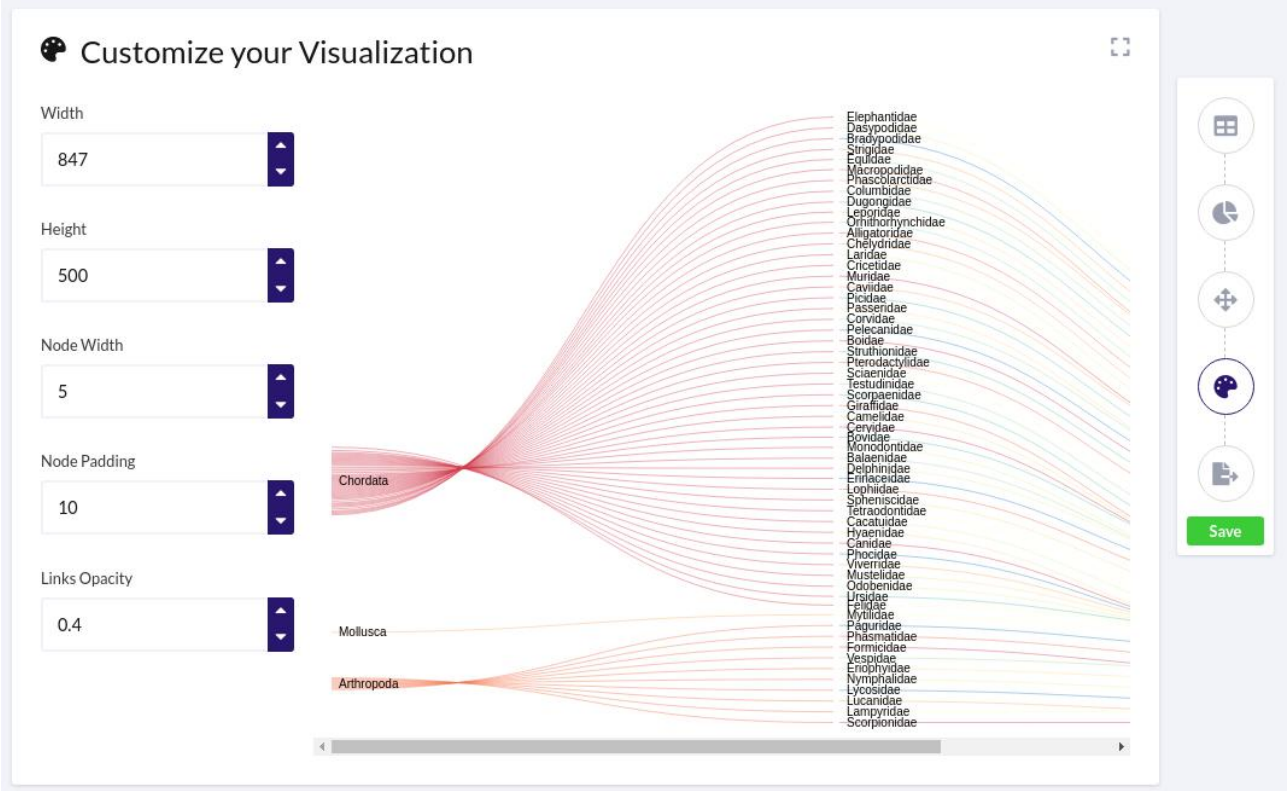


Рисунок 5.4 – Форма редагування зовнішнього вигляду створеної діаграми

Створене графічне представлення можна завантажити або експортувати для безпосередньої інтеграції у код HTML-сторінки (рисунок 5.5).

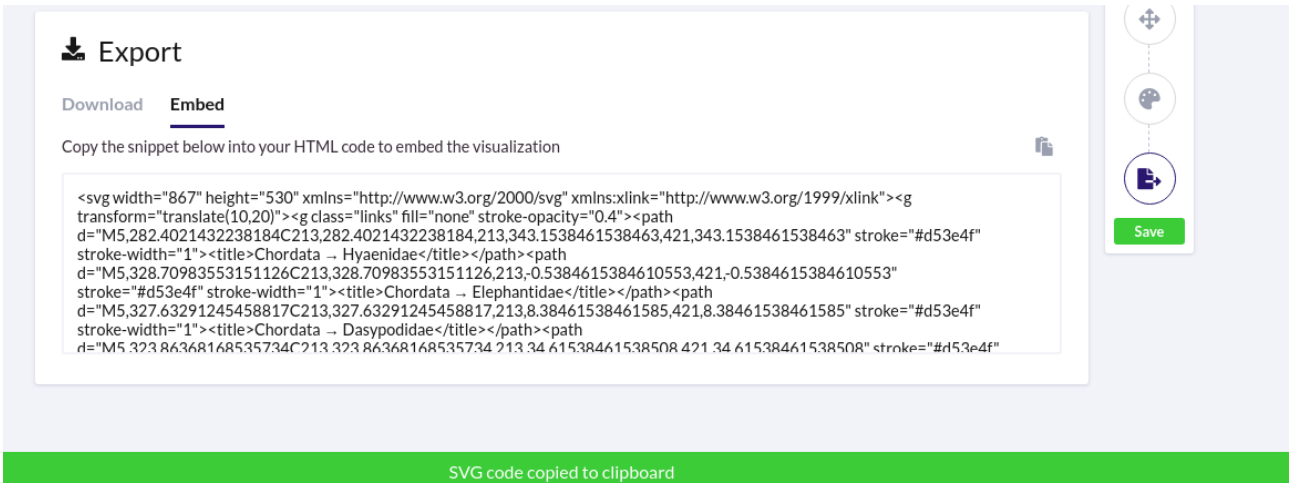


Рисунок 5.5 – Форма експорту створеної діаграми

Проектами, до яких має доступ конкретний користувач, легко керувати на високому рівні абстракції (рисунок 5.6).

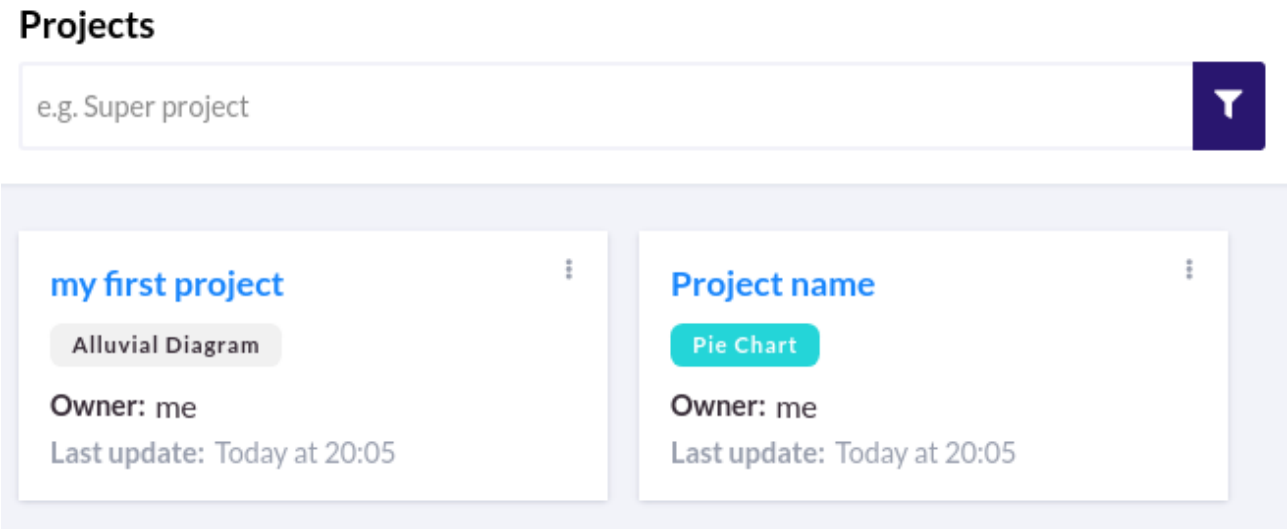


Рисунок 5.6 – Форма управління проектами

До таких операцій належить надання доступу іншим користувачам до окремих власних проектів. Для цього необхідно надати адресу електронної пошти іншого користувача (рисунок 5.7).

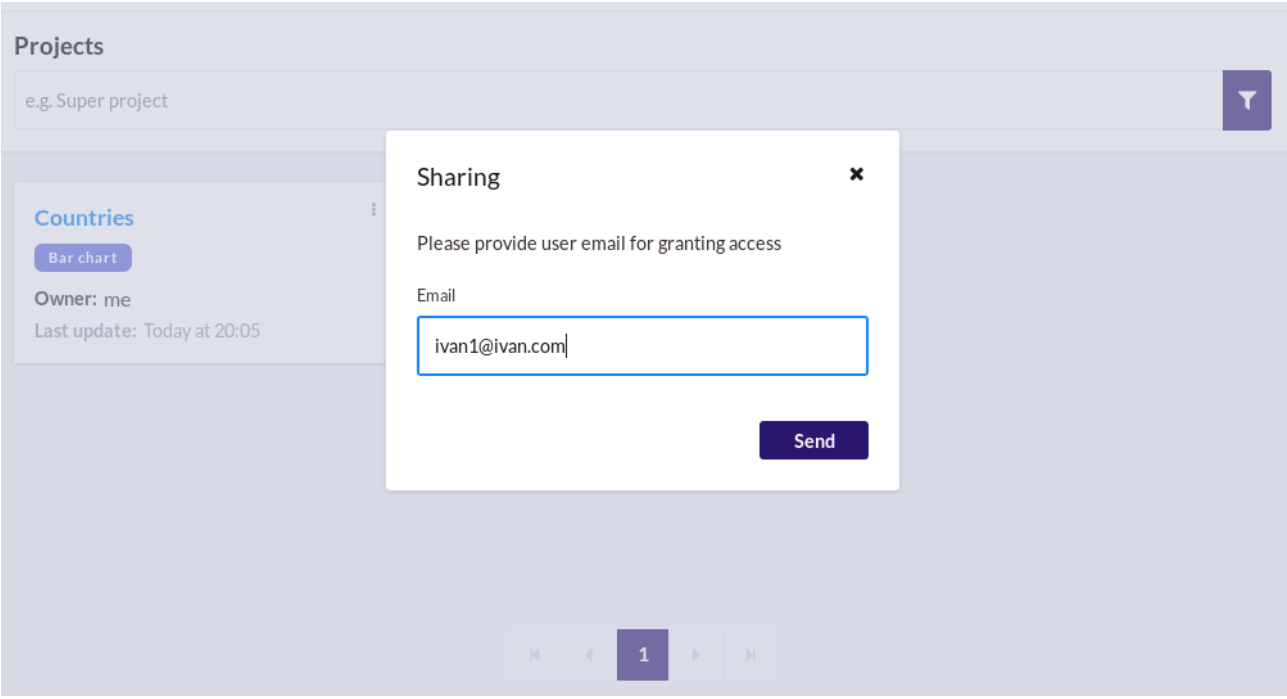


Рисунок 5.7 – Форма створення спільного доступу до проекту

Якщо користувач отримає можливість спільного доступу до проекту іншого користувача, він побачить дані про даний проект на своїй формі управління проектами (рисунок 5.8).

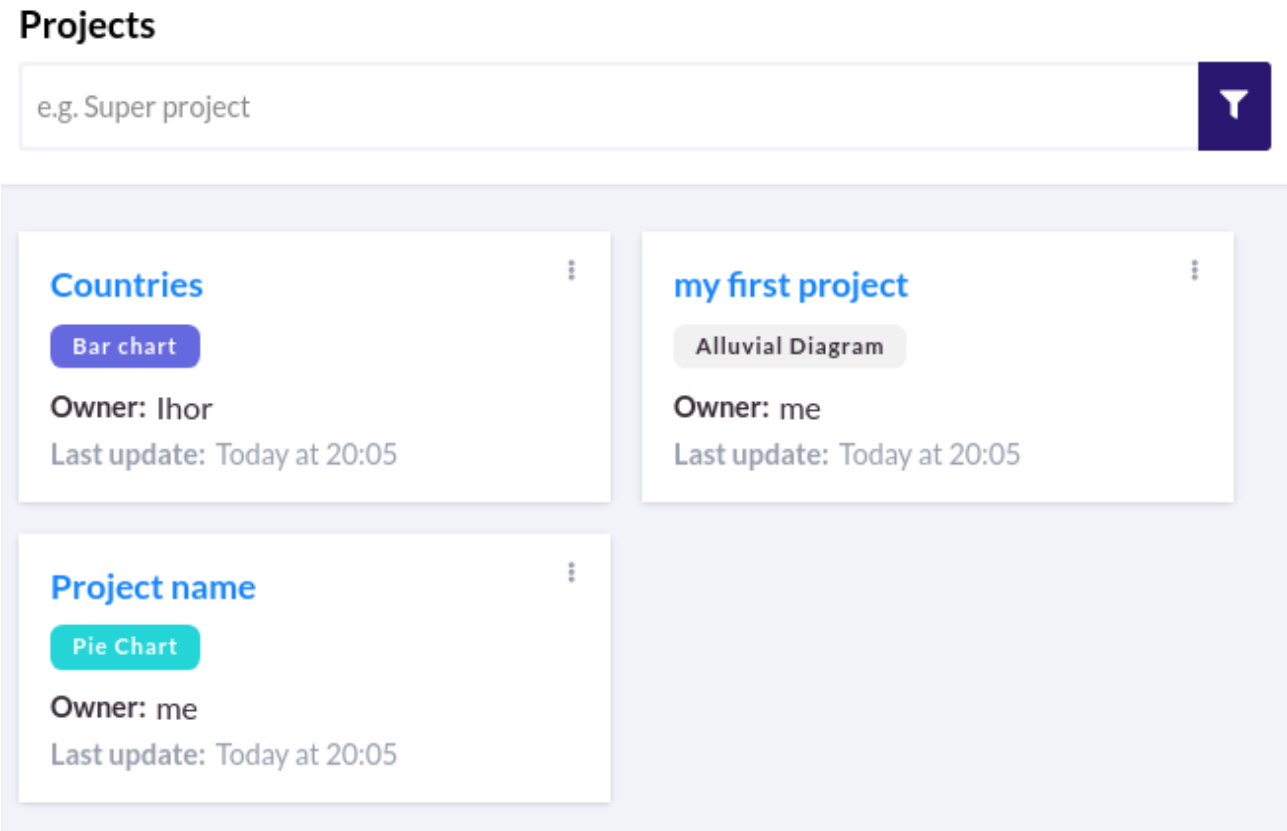


Рисунок 5.8 – Результати надання спільного доступу до проекту іншому користувачеві

5.3 Висновки до розділу

Було описано основні події типового сценарію роботи користувача із системою, які відображають внутрішній функціонал програмного комплексу, надаючи прості та потужні рішення проблем графічної візуалізації даних та надання спільного доступу до своїх проектів іншим користувачам.

ВИСНОВКИ

У ході детального аналізу предметної області візуалізації числових даних за допомогою різних типів графіків та діаграм було виявлено суттєві недоліки існуючи програмних засобів, які реалізують даний процес.

Було запропоновано створення нової програмної системи, що дозволить легко будувати різні типи графіків та діаграм, якими можна буде зручно ділитися з іншими користувачами.

Було проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір веб-додатку для браузера, що дає змогу не прив’язуватися до програмного забезпечення користувача, та методу розробки на основі шаблону MVP. Це дало змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використанні кінцевими користувачами програмного засобу.

За результатами виконання тестових завдань була підтверджена коректність отриманих результатів, отже створена система відповідає поставленим вимогам.

Користувачами системи можуть бути працівники компаній та пересічні споживачі Інтернет-ресурсів, які бажають побудувати діаграми для різних цілей та мати можливість ділитися ними з іншими користувачами системи. Веб-додаток може бути використано на будь-якому персональному комп’ютері та на будь-якій операційній системі.

Отже, створена програмна система є ефективним рішенням для широко спектру задач візуалізації числових даних у різних форматах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How to make an alluvial diagram [Електронний ресурс] — Режим доступу: https://rawgraphs.io/learning/how-to-make-an-alluvial-diagram/.

2. Фримен А. — Angular для профессионалов [Електронний ресурс]. — 2018. — Режим доступу: http://zakazknig.com.ua/adam-frimen-angular-dlya-professionalov.

3. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: http://www.rusdoc.ru/articles/18358/.

4. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009 — Режим доступу: https://habrahabr.ru/post/53536/.

5.Сухов К. — Node.js. Путеводитель по технологии [Електронний ресурс]. — 2015. — Режим доступу: https://rozetka.com.ua/70309688/p70309688/.

6. Этан Б. — Изучаем JavaScript. Руководство по созданию современных веб-сайтов [Електронний ресурс]. — 2016. – Режим доступу: https://www.bambook.com/book/rus/izuchaem-javascript-rukovodstvo-po-sozdaniyu-sovremennyih-veb-say-1694770.

7. Моргунов Евгений PostgreSQL. Основы языка SQL. Учебное пособие, 2016. — 336 с.

8. Болье А. — Learning SQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.

ДОДАТОК 1

Мульти графічна візуалізація числових даних та їх обмін між користувачами

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51192_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ51192_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ51192_19Б 12-1	app/server	Модулі серверної частини системи
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ51192_19Б 12-2	app/client	Модулі клієнської частини системи

ДОДАТОК 2

Сервіси клієнтської частини

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51192_19Б 12-1

Аркушів 4

Київ – 2019

```
                                        //Імпорт зовнішніх модулів
import { Injectable } from '@angular/core';
import { DimensionOption } from '@app/models/chart.model';
import { DimensionColumnMap } from '@app/models/chart.model';
import { v4 } from 'uuid';
import { SchemeID } from '@app/models/normalizr.model';
//Оголошення класу
@Injectable()
export class ChartService {
        transformDimensions(
                userChartId: SchemeID,
                dimensions: DimensionOption[]
        ): DimensionColumnMap[] {
                return dimensions.map(el => ({
                        id: `${userChartId}-${el.id}`,
                        columnIds: []
                }));
        }
//Функція створення діаграми
        createChart({
                chartTypeId,
                datasetId,
                customizeSettings,
                dimensionSettings
        }) {
                const id = v4();
                return {
                        id,
```

```
                chartTypeId,
                datasetId,
                customizeSettings,
                dimensionSettings: this.transformDimensions(id, dimensionSettings)
            };
        }
}
import { Injectable } from '@angular/core';
import { Observable, of } from 'rxjs';
import { v4 } from 'uuid';
import {
    DatasetTable,
    Dataset,
    DatasetColumn,
    DatasetData
} from '@app/models/dataset.model';
//Оголошення іншого класу
@Injectable()
export class DatasetService {
    transformDatasets(
        datasets: DatasetTable[] = [],
        isDraft: boolean = true
    ): Dataset<DatasetColumn[], DatasetData[][]>[] {
        return datasets.map(
            d =>
                ({
                    id: d.id,
                    source: {
```

```
                    ...d
                },
                modified: {
                    id: d.id,
                    columns: [...d.columns],
                    data: d.data.map((r, rI) =>
                        r.map((c, cI) => ({
                            id: `${rI}-${d.columns[cI].id}-${d.id}`,
                            value: c
                        }))
                    )
                },
                isDraft
            } as Dataset<DatasetColumn[], DatasetData[][]>)
        );
    }//Створення набору даних користувача
    createDataset(
        columns: DatasetColumn[],
        data: any[][]
    ): Observable<DatasetTable> {
        const dataset = {
            id: v4(),
            columns,
            data
        };
        return of(dataset);
    }
}
```

ДОДАТОК 3

Сервіси клієнтської частини

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51192_19Б 12-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Модуль поєднує серверну та клієнтську частини системи.

Методи сервісів надають компонентам доступ до бізнес-логіки створення діаграм та наборів даних, управління ними.

Модуль було створено у середовищі Visual Studio Code на мові програмування TypeScript.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код модулю міститься в межах пакету client/src/app/services, який реалізує взаємодію клієнтської частини програмного комплексу із серверною.

Модуль відіграє ключову роль у функціонуванні системи, оскільки надає компонентам доступ до бізнес-логіки.

Програма була написана мовою TypeScript з використанням можливостей фреймворку Angular.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Сервіси реалізують наступний функціонал:

- створення наборів даних;
- завантаження наборів даних та їх зміни;
- створення діаграм;
- збереження діаграм;
- можливості подальшої зміни конфігурації діаграм.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Модуль реалізовано у формі двох класів:

1) ChartService, що містить операції для забезпечення взаємодії користувача з діаграмами;
2) DatasetService, що містить операції для забезпечення взаємодії користувача з наборами даних, які використовуються для побудови та редагування діаграм.

-7-

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Програмний код модулю було написано у середовищі розробки Visual Studio Code.

Копія коду, як і всього проекту, наявна у системі GitHub.

Синтаксис модулю відповідає вимогам мови TypeScript.

Використовуються стандартні та сторонні модулі системи фреймворку Angular.

-8-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

— набір даних від користувача;

— діаграма користувача.

Вихідними даними є:

— набір даних на сервері;

— діаграма на сервері.